

AAX SDK

2.6.1

Generated by Doxygen 1.9.6

1 Main Page	1
1.1 Welcome to AAX	1
1.1.1 The Basics	1
1.1.2 More Topics	2
1.1.3 Test Tools & Utilities	2
1.1.4 Supplemental Information	2
1.2 SDK Folder Hierarchy	2
1.3 Contacting Avid	3
1.4 Licensing	3
2 Todo List	5
3 Host Compatibility Notes	9
4 Legacy Porting Notes	15
5 Deprecated List	19
6 Not Used by AAX Plug-Ins	21
7 Module Index	23
7.1 Manual	23
8 Namespace Index	25
8.1 Namespace List	25
9 Hierarchical Index	27
9.1 Class Hierarchy	27
10 Class Index	31
10.1 Class List	31
11 File Index	37
11.1 File List	37
12 Module Documentation	43
12.1 AAX SDK Manual	43
12.1.1	43
12.1.2 Welcome to AAX	43
12.1.2.1 The Basics	43
12.1.2.2 More Topics	44
12.1.2.3 Test Tools & Utilities	44
12.1.2.4 Supplemental Information	44
12.1.3 SDK Folder Hierarchy	44
12.1.4 Contacting Avid	45
12.1.5 Licensing	45

12.2 Getting Started with AAX	47
12.2.1 Contents	47
12.2.2 Welcome	47
12.2.3 Quick Start	47
12.2.4 AAX Design Overview	48
12.2.4.1 Architecture Philosophy	48
12.2.4.2 Design Attributes	48
12.2.4.3 Component Structure	49
12.2.4.4 Algorithm	49
12.2.4.5 Data Model	50
12.2.4.6 GUI Interface	50
12.2.4.7 Describe	50
12.2.4.8 Controller	51
12.2.5 DemoGain Example	51
12.2.5.1 AAX Plug-In Parameters	51
12.2.5.2 Data Model: Create Your Parameter	52
12.2.5.3 Algorithm: Add coefficients to the algorithm's context structure	52
12.2.5.4 Describe: Connect the parameter throughout the plug-in	53
12.2.5.5 GUI: Add a control	54
12.2.6 Next Steps	54
12.3 Core AAX Interface	55
12.3.1	55
12.4 Description callback	56
12.4.1	56
12.4.2 On this page	56
12.4.3 About the Describe callback and AAX descriptor interfaces	57
12.4.4 Top level: Collection	58
12.4.5 Middle level: Effects	59
12.4.5.1 Registering multiple Effects	59
12.4.6 Lowest level: Algorithm components	60
12.4.6.1 Algorithm callback properties	60
12.4.7 Checking Results	61
12.4.7.1 Summary	61
12.4.7.2 The Problem	61
12.4.7.3 The Solution	62
12.4.7.4 Handling Errors and Managing Control Flow	62
12.4.8 Describe Validation	64
12.4.8.1 Validation with DSH	64
12.4.8.2 Validation with Pro Tools	64
12.4.9 Additional Topics	64
12.4.10 Function Documentation	65
12.4.10.1 AAXRegisterPlugin()	65

12.4.10.2 GetEffectDescriptions()	65
12.5 Real-time algorithm callback	66
12.5.1 On this page	66
12.5.2 Algorithm definition	66
12.5.3 Algorithm memory management	67
12.5.4 Communicating with the algorithm	68
12.5.5 Algorithm initialization	68
12.5.5.1 Private data initialization	68
12.5.5.2 Optional initialization callback	69
12.5.6 Algorithm processing	69
12.5.7 Persistent algorithm memory	69
12.5.7.1 Private memory characteristics	69
12.5.7.2 Private data port registration	70
12.5.7.3 Private data initialization	70
12.5.7.4 Private data communication	70
12.5.8 Example algorithm callback	70
12.5.9 Port Types and Behavior	71
12.5.9.1 Standard message input	71
12.5.9.2 Internal state storage	71
12.5.9.3 Metering output	71
12.5.9.4 Environment variable retrieval	72
12.5.9.5 Other functionality enhancement	72
12.5.10 Additional Information	72
12.6 Data model interface	72
12.6.1	72
12.6.2 Related classes	73
12.7 GUI interface	74
12.7.1	74
12.8 AAX communication protocols	76
12.8.1 Communication with the C++ interface objects	76
12.8.1.1 Direct host communication	76
12.8.1.2 Custom data blocks	76
12.8.1.3 Notifications	77
12.8.1.4 Direct pointer sharing	77
12.8.2 Communication with the real-time algorithm	77
12.8.2.1 Data packets	77
12.8.2.2 Host-managed context fields	77
12.8.2.3 Direct data transfers	78
12.9 AAX Format Specification	78
12.9.1 .aaxplugin Directory Structure	78
12.9.2 Required Symbols	79
12.10 Additional AAX features	80

12.10.1	80
12.11 Direct data access interface	81
12.11.1	81
12.11.2 Convenience class	82
12.11.3 Private data access interface	82
12.11.4 Communicating with other modules	82
12.12 Offline processing interface	83
12.12.1	83
12.13 Hybrid Processing architecture	84
12.13.1	84
12.13.2 Overview of Hybrid	84
12.13.3 Implementing Hybrid processing	84
12.13.4 Additional information	85
12.13.4.1 Parameter update timing	85
12.13.4.2 Host support and alternatives	85
12.13.5 Function Documentation	86
12.13.5.1 RenderAudio_Hybrid()	86
12.13.5.2 GetHybridSignalLatency()	86
12.14 MIDI	87
12.14.1 Midi Overview	87
12.14.2 MIDI node types	87
12.14.3 Adding MIDI functionality to a plug-in	88
12.14.4 Using MIDI in a plug-in algorithm	88
12.14.5 Accessing MIDI in the plug-in data model	89
12.15 Plug-in meters	90
12.15.1 Overview of metering in AAX	90
12.15.2 Adding meters to an Effect	90
12.15.2.1 Customizing meter behavior	90
12.15.3 Reporting meter values	91
12.15.4 Displaying meter values	91
12.15.5 Alternatives	92
12.16 Sidechain Inputs	92
12.16.1 Overview of Sidechain Inputs	92
12.16.2 Adding a Sidechain Input to an Effect	92
12.17 Auxiliary Output Stems	93
12.17.1 Overview of Auxiliary Output Stems in AAX	93
12.17.2 Implementing Auxiliary Output Stems	94
12.18 Direct Memory Access	94
12.18.1 On this page	95
12.18.2 DMA facility overview	95
12.18.3 DMA transfer modes	95
12.18.4 Registering for DMA transfers	95

12.18.5 DMA restrictions	95
12.18.6 Additional information	96
12.19 Background processing callback	96
12.19.1 On this page	96
12.19.2 Background thread description	96
12.19.3 Restrictions and limitations of background threads	96
12.19.4 Background thread performance characteristics on DSP systems	97
12.19.5 Background thread memory management	97
12.19.6 Additional information	97
12.20 EQ and Dynamics Curve Displays	98
12.20.1	98
12.20.2 Enumeration Type Documentation	100
12.20.2.1 AAX_ECureType	100
12.20.3 Function Documentation	101
12.20.3.1 GetCurveData()	101
12.20.3.2 GetCurveDataMeterIds()	102
12.20.3.3 GetCurveDataDisplayRange()	102
12.21 Task agent interface	103
12.21.1	103
12.21.2 Communicating with other modules	104
12.21.3 Enumeration Type Documentation	104
12.21.3.1 AAX_TaskCompletionStatus	104
12.22 AAX Library features	105
12.22.1	105
12.23 Parameter Manager	105
12.23.1	105
12.23.2 Parameter concepts	106
12.23.2.1 Parameter value domains	106
12.23.2.2 Taper	107
12.23.2.3 Delegates	107
12.23.2.4 Model-View-Controller	107
12.24 Taper delegates	108
12.24.1	108
12.25 Display delegates	109
12.25.1	109
12.25.2 Display delegate decorators	109
12.25.2.1 Display delegate decorator implementation	109
12.25.2.2 Decibel decorator example	110
12.26 Display delegate decorators	110
12.26.1	110
12.27 Additional Topics	111
12.27.1	111

12.28 Real-time performance	112
12.28.1 Things NOT To Do In An Audio Plug-In Render Callback	112
12.28.2 Things To Do In An Audio Plug-In Render Callback	113
12.28.3 Good Resources And Examples	113
12.29 Parameter automation	113
12.29.1 On this page	114
12.29.2 Overview	114
12.29.3 Plug-in elements used for automation	114
12.29.3.1 Defining automatable parameters	115
12.29.4 Advanced automation topics	115
12.30 Parameter updates	115
12.30.1	115
12.31 Parameter update timing	116
12.31.1 On this page	116
12.31.2 Timeline Locations	117
12.31.3 Coordinating the data model and algorithm	117
12.31.3.1 A closer look at the AAX packet delivery system	117
12.31.4 Fixing timing issues due to shared data	118
12.31.4.1 Monolithic plug-ins	118
12.31.4.2 How to resolve timing errors	119
12.31.4.3 Additional considerations	120
12.31.5 Determining the absolute timestamp for a parameter update	121
12.31.5.1 Obtaining timeline information	121
12.31.5.2 Determining the timeline position of a parameter update	121
12.32 Token protocol	123
12.32.1 On this page	123
12.32.2 An Introduction to Tokens	124
12.32.2.1 Touch	124
12.32.2.2 Set	125
12.32.2.3 Update	125
12.32.3 Basic Token Operation	126
12.32.3.1 User Editing	126
12.32.3.2 Automation Playback	127
12.32.3.3 Chunk Restoring	127
12.33 Basic parameter update sequences	127
12.33.1 On this page	127
12.33.1.1 Notes on threading for these sequences	128
12.33.2 User-generated update	128
12.33.2.1 High-level interface calls and events	128
12.33.2.2 Detailed sequence for default implementation	128
12.33.2.3 Updates from control surfaces	129
12.33.3 Automation playback	130

12.33.4 Initialization	130
12.34 Linked parameters	131
12.34.1 On this page	132
12.34.2 Basics of Linked Parameters	132
12.34.2.1 Basic considerations for parameter linking	132
12.34.2.2 Defining proper linked parameter behavior	133
12.34.3 Linked Parameter Operation	134
12.34.3.1 User Editing	134
12.34.3.2 Automation Playback	135
12.34.3.3 Chunk Restoring	136
12.34.4 Changing Tapers	136
12.35 Linked parameter update sequences	136
12.35.1 On this page	137
12.35.1.1 Notes on threading for these sequences	137
12.35.2 User-generated update	137
12.35.2.1 High-level interface calls and events	138
12.35.2.2 Detailed interface calls and events	138
12.35.3 Update from automation playback	139
12.36 Plug-in type conversion	140
12.36.1 About this specification	140
12.36.2 Terminology	141
12.36.3 Scope of this specification	142
12.36.4 Topological constraints	142
12.36.5 Implicit conversions	142
12.36.6 Explicit conversions	143
12.36.7 Type deprecation	143
12.37 The Avid Component Framework (ACF)	144
12.37.1	144
12.37.2 More details	145
12.37.3 ACF interfaces in AAX	145
12.37.4 Using ACF interfaces	146
12.37.4.1 Host-provided interfaces	146
12.37.4.2 Plug-in interfaces	146
12.37.5 Interface versioning in AAX	147
12.38 ACF Elements	149
12.38.1	149
12.39 AAX Host Guides	150
12.39.1	150
12.40 Pro Tools Guide	151
12.40.1 Contents	151
12.40.2 About this document	152
12.40.3 Processing modes	152

12.40.3.1 Real-time processing	152
12.40.3.2 Non-real-time processing (AudioSuite)	153
12.40.3.3 Multichannel and Multi-Mono	153
12.40.4 Requirements for AAX plug-in compatibility with Pro Tools	153
12.40.4.1 Install directories	154
12.40.4.2 Plug-in name and file structure	154
12.40.4.3 Digital signature	154
12.40.5 Audio Engine Behavior and Features	156
12.40.5.1 Plug-in loading and AAE initialization	156
12.40.5.2 Plug-in initialization	156
12.40.5.3 Run-time processing behavior	157
12.40.6 Basic plug-in operation	158
12.40.6.1 Configuration management	158
12.40.6.2 Plug-in activation and deactivation	159
12.40.6.3 Plug-in bypass	159
12.40.6.4 Presets and settings management	159
12.40.6.5 Modifier key behavior	161
12.40.7 Optional plug-in features	162
12.40.7.1 Audio management features	162
12.40.7.2 Plug-in categories	164
12.40.7.3 Advanced non-real-time processing	165
12.40.8 Using the Pro Tools Scripting SDK with AAX	166
12.40.9 Debugging AAX plug-ins	166
12.40.9.1 Debugging within Pro Tools	166
12.40.9.2 DigiShell	167
12.40.9.3 DigiTrace	167
12.40.10 Troubleshooting common AAX plug-in failures	167
12.40.11 Using DigiOptions	168
12.40.11.1 Useful DigiOptions	168
12.40.12 Compatibility Notes	170
12.41 Media Composer Guide	170
12.41.1 Contents	170
12.41.2 About this document	170
12.41.3 Processing modes	171
12.41.3.1 Non-real-time processing (AudioSuite)	171
12.41.3.2 Real-time processing	172
12.41.4 Compatibility requirements	174
12.41.4.1 Install directories	174
12.41.4.2 Plug-in name and file structure	175
12.41.5 AAX feature support in Media Composer	175
12.41.5.1 Processing configurations	175
12.41.5.2 Preset management	176

12.41.5.3 Unsupported features	177
12.41.5.4 Additional feature support notes	177
12.41.6 Additional Information	177
12.41.6.1 Audio Engine features and behavior	177
12.41.6.2 Debugging AAX plug-ins in Media Composer	178
12.42 HDX DSP Guide	178
12.42.1 Contents	178
12.42.2 Overview of TI DSP Algorithms in AAX	178
12.42.3 Getting Started with HDX DSP	179
12.42.4 The HDX DSP Platform	179
12.42.4.1 DSP characteristics: instruction processing	179
12.42.4.2 DSP characteristics: audio buffers	179
12.42.4.3 DSP characteristics: memory	180
12.42.4.4 System characteristics: DSP/host data transfers	180
12.42.4.5 TI Shell characteristics: Memory allocation	181
12.42.4.6 TI Shell characteristics: Data packet services	182
12.42.4.7 TI Shell characteristics: Instance allocation	183
12.42.4.8 Additional TI Shell services	184
12.42.5 Requirements for HDX DSP Plug-Ins	184
12.42.5.1 Plug-in description	184
12.42.5.2 Performance measurement and reporting	185
12.42.5.3 Plug-in compilation and packaging	186
12.42.6 TI Development Tools	187
12.42.6.1 Code Composer Studio	187
12.42.6.2 The TMS320C6000 C++ compiler	191
12.42.6.3 DigiShell test tool (DSH)	192
12.42.6.4 Hardware Debugging	193
12.42.6.5 Tracing	195
12.42.6.6 Testing in Pro Tools	196
12.42.7 Common Issues with TI Development	196
12.42.7.1 Data structure compatibility	196
12.42.8 TI Optimization Guide	200
12.42.8.1 Optimization quick start	200
12.42.8.2 Compiler and linker options	201
12.42.8.3 The load-update-store pattern	203
12.42.8.4 Case study: IIR filter implementation on TI 672x DSPs	204
12.42.8.5 Understanding CGTools-generated ASM files	205
12.42.8.6 C keywords	207
12.42.8.7 Data types	209
12.42.8.8 Case study: Efficient parameter smoothing at single and double precision	211
12.42.8.9 Refactoring conditionals and branches	212
12.42.8.10 Case study: pipeline refactoring in Avid's EQ3 and Dyn3 plug-ins	214

12.42.8.11 Case study: Additional optimization lessons from EQ3 and Dyn3	217
12.42.8.12 Optimization on the HDX platform	218
12.42.8.13 Code Composer Studio optimization tools	219
12.42.9 Error Codes	219
12.42.9.1 -138xx: DHM Core DSP errors	220
12.42.9.2 -140xx: AAX Host errors	220
12.42.9.3 -141xx: TI System errors	221
12.42.9.4 -142xx: DIDL errors	222
12.42.9.5 -144xx: HDX hardware errors	222
12.42.9.6 -145xx: DHM isochronous audio engine errors	222
12.42.9.7 -30xxx: Dynamically-generated error codes	223
12.43 Page Table Guide	224
12.43.1 Contents	224
12.43.2 Introduction	224
12.43.2.1 Control Surfaces Overview	224
12.43.2.2 Page Tables Overview	224
12.43.3 Avid Control Surfaces	225
12.43.3.1 EUCON	225
12.43.3.2 Avid C 24 and ICON	226
12.43.3.3 VENUE	228
12.43.4 Plug-In Page Table Guidelines	229
12.43.4.1 General Guidelines	229
12.43.5 Avid Center Section Page Tables	230
12.43.5.1 Center Section Page Table Guidelines	230
12.43.5.2 Center Section Parameter Mapping to Single-Column/Row Layouts	235
12.43.5.3 Center Section Parameter Mapping in S6 Expand Mode	246
12.43.5.4 Center Section Parameter Mapping on VENUE S3L-X	246
12.43.6 EUCON Page Tables	247
12.43.6.1 Specification	247
12.43.6.2 Types	248
12.43.6.3 Conventions	248
12.43.6.4 Requirements	248
12.43.7 Implementing Page Tables	248
12.43.7.1 Page table XML specification	248
12.43.7.2 Parameter identifiers	251
12.43.7.3 Creating page tables using the AAX Plug-In Page Table Editor	252
12.43.7.4 Verifying Page Table Layouts: The Hidden Pop-Up Menu	253
12.43.7.5 Control Highlighting Scheme	254
12.43.7.6 Control Numbering Layouts	254
12.43.7.7 Alphanumeric Displays	255
12.43.7.8 ProControl Display	256
12.43.8 Appendix A. Get Parameter Value Info	257

12.43.8.1 Overview	257
12.43.8.2 Implementation	257
12.44 DigiTrace Guide	260
12.44.1 On this page	260
12.44.2 What is DigiTrace?	260
12.44.2.1 What does DigiTrace do?	260
12.44.3 DigiTrace quick start guide	261
12.44.3.1 Find and decrypt DigiTrace log files	261
12.44.3.2 Configure DigiTrace for AAX plug-in logging	261
12.44.3.3 Configure DigiTrace for plain-text output	262
12.44.3.4 Add tracing to a plug-in	262
12.44.4 DigiTrace log files	262
12.44.4.1 Where are DigiTrace log files stored?	262
12.44.4.2 Monitoring DigiTrace logs	263
12.44.4.3 Log file formatting	263
12.44.5 Configuring DigiTrace	264
12.44.5.1 Trace facilities	264
12.44.5.2 Trace priorities	264
12.44.5.3 Useful DigiTrace facilities	265
12.44.6 Bonus features	265
12.44.6.1 Real-time AAE performance logging with DigiTrace	265
12.44.6.2 Adding signposts to the DigiTrace log at run-time	266
12.44.7 Adding traces to an AAX plug-in	266
12.44.7.1 Basic AAX logging	266
12.44.7.2 Advanced DigiTrace logging features	267
12.44.7.3 Security concerns	268
12.44.8 Advanced DigiTrace configuration	269
12.44.8.1 Configuration command format	269
12.44.8.2 Advanced configuration commands	269
12.44.8.3 Dynamically changing the DigiTrace configuration	270
12.44.9 Compatibility	270
12.44.10 Additional Information	271
12.44.10.1 Confidentiality	271
12.45 DSH Guide	271
12.45.1 Contents	271
12.45.2 What is DSH and how it works	271
12.45.3 Basic set of commands of the DAE dish	272
12.45.3.1 Loading plug-ins in DSH	272
12.45.3.2 Working with HDX card from DSH	273
12.45.3.3 DAE dish tips	273
12.45.4 Basic plug-in tests	273
12.45.4.1 Cycle count performance test	274

12.45.4.2 Cancellation test	275
12.45.5 Debugging and tracing in DSH	276
12.45.6 Scripting interface and batch profiling	276
12.46 DTT Guide	277
12.46.1 Contents	277
12.46.2 What is DTT	277
12.46.3 How to run tests and suites in DTT	278
12.46.4 Writing DTT scripts	278
12.46.4.1 Describing and using input arguments of the script	278
12.46.4.2 Writing body of the script	279
12.46.5 Logging in DTT and debugging DTT scripts	279
12.46.5.1 Interactive mode	279
12.46.6 Working with DTT test suites	280
12.46.6.1 Autogeneration of the suites	280
12.47 Extensions	281
12.47.1	281
12.48 GUI Extensions	282
12.48.1 About the SDK's GUI Extensions	282
12.48.2 Notes	283
12.49 Monolithic VIs and Effects	283
12.50 Other Extensions	284
12.50.1	284
12.50.2 Function Documentation	284
12.50.2.1 AsStringMIDIStream_Debug()	284
12.50.2.2 GetPathToPlugInBundle()	284
12.51 Supplemental Information	285
12.51.1	285
12.52 Troubleshooting	286
12.52.1 Contents	286
12.52.2 Plug-In Fails to Load in Shipping Pro Tools	286
12.52.3 Plug-In Causes Audio Streaming Errors	288
12.53 Distributing Your AAX Plug-In	290
12.53.1 Contents	290
12.53.2 The finishing touches	290
12.53.2.1 Check and finalize page tables	290
12.53.2.2 Create factory presets	290
12.53.2.3 Sign your plug-in	291
12.53.3 Building your plug-in installer	291
12.53.3.1 Installing Track Presets	292
12.53.4 Testing your plug-in	293
12.53.5 Selling your plug-in	293
12.53.5.1 Avid Marketplace	293

12.53.5.2 In-App Purchase	293
12.54 AAX Interfaces	294
12.54.0.1 Interfaces Implemented by the AAX Host	294
12.54.0.2 Interfaces Implemented by the AAX Plug-In	295
12.54.0.3 Interfaces internal to the AAX SDK	295
12.55 Host Support	295
12.55.1 Host Support	296
12.55.1.1 Platform Support	296
12.55.1.2 Describe Interfaces	296
12.55.1.3 Run-Time Interfaces	297
12.55.1.4 Features	297
12.55.2 Host Compatibility Notes	298
12.56 Known Issues	302
12.56.1 Contents	302
12.56.2 Known Issues in the AAX SDK	302
12.56.2.1 AAXSDK-897	302
12.56.2.2 AAXSDK-851	303
12.56.2.3 AAXSDK-832	303
12.56.2.4 AAXSDK-708	303
12.56.2.5 AAXSDK-705	303
12.56.2.6 AAXSDK-663	303
12.56.2.7 AAXSDK-599	303
12.56.2.8 AAXSDK-561 / PT-232159	303
12.56.2.9 AAXSDK-533	304
12.56.2.10 AAXSDK-514	304
12.56.2.11 AAXSDK-321	304
12.56.2.12 AAXSDK-271	304
12.56.2.13 AAXSDK-186	304
12.56.2.14 AAXSDK-162	305
12.56.2.15 AAXSDK-16	305
12.56.2.16 AAXSDK-14	305
12.56.2.17 AAXSDK-13 / AAX-579 / PTSW-158381	305
12.56.2.18 AAXSDK-11 / AAX-581 / PTSW-158348	305
12.56.2.19 AAXSDK-10 / AAX-580 / PTSW-154083	305
12.56.2.20 AAXSDK-6 / AAX-646	306
12.56.2.21 AAXSDK-5	306
12.56.2.22 AAXSDK-2 / AAX-648	306
12.56.2.23 AAX-582 / PTSW-157726	306
12.56.2.24 AAX-585 / PTSW-157451	306
12.56.2.25 AAX-578 / PTSW-158310	306
12.56.3 Known Issues in Pro Tools	307
12.56.3.1 PT-307193	307

12.56.3.2 PT-305352	307
12.56.3.3 PT-303482	307
12.56.3.4 PT-299906	307
12.56.3.5 PT-297802	307
12.56.3.6 PT-290588	307
12.56.3.7 PT-284916	308
12.56.3.8 PT-282946	308
12.56.3.9 PT-278282	308
12.56.3.10 PT-276280	308
12.56.3.11 PT-274717	308
12.56.3.12 PT-271830	308
12.56.3.13 PT-263909	308
12.56.3.14 PT-263859	309
12.56.3.15 PT-261394	309
12.56.3.16 PT-258560 / PT-256919	309
12.56.3.17 PT-258394	309
12.56.3.18 PT-257213	309
12.56.3.19 PT-256704	309
12.56.3.20 PT-255800	309
12.56.3.21 PT-255408	310
12.56.3.22 PT-254203	310
12.56.3.23 PT-254118 / PT-275441 / PT-279941	310
12.56.3.24 PT-254103	310
12.56.3.25 PT-250751	310
12.56.3.26 PT-249791	310
12.56.3.27 PT-249790	310
12.56.3.28 PT-248000	311
12.56.3.29 PT-245693	311
12.56.3.30 PT-243211	311
12.56.3.31 PT-237857	311
12.56.3.32 PT-236755	311
12.56.3.33 PT-235831	311
12.56.3.34 PT-235333	311
12.56.3.35 PT-234681	312
12.56.3.36 PT-233726	312
12.56.3.37 PT-233176	312
12.56.3.38 PT-232678 / PT-236755	312
12.56.3.39 PT-232403	312
12.56.3.40 PT-232159	312
12.56.3.41 PT-230327	312
12.56.3.42 PT-230290	313
12.56.3.43 PT-230288	313

12.56.3.44 PT-229026	313
12.56.3.45 PT-227655	313
12.56.3.46 PT-227173	313
12.56.3.47 PT-226559	313
12.56.3.48 PT-225763	313
12.56.3.49 PT-223581	314
12.56.3.50 PT-218545	314
12.56.3.51 PT-218486	314
12.56.3.52 PT-210904 / VSW-14216	314
12.56.3.53 PT-206995	314
12.56.3.54 PT-206541	314
12.56.3.55 PT-206161	315
12.56.3.56 PT-205610	315
12.56.3.57 PT-203420	315
12.56.3.58 PT-202345	315
12.56.3.59 PTSW-200437 / PTSW-197598	315
12.56.3.60 PTSW-197651 / PT-218405	315
12.56.3.61 PTSW-197601 / PT-218459	315
12.56.3.62 PTSW-197593 / PT-218480	316
12.56.3.63 PTSW-197540	316
12.56.3.64 PTSW-197472	316
12.56.3.65 PTSW-197471	316
12.56.3.66 PTSW-197468 / PT-218460	316
12.56.3.67 PTSW-197431 / PT-218414	316
12.56.3.68 PTSW-197075	316
12.56.3.69 PTSW-196772 / PT-218423	317
12.56.3.70 PTSW-196604	317
12.56.3.71 PTSW-196428 / PT-218488	317
12.56.3.72 PTSW-195316 / PT-218485	317
12.56.3.73 PTSW-195257	317
12.56.3.74 PTSW-195256 / PT-218429	317
12.56.3.75 PTSW-195209 / PT-218474	317
12.56.3.76 PTSW-195113	318
12.56.3.77 PTSW-194698 / PT-218478	318
12.56.3.78 PTSW-194231 / PT-218434	318
12.56.3.79 PTSW-193646	318
12.56.3.80 PTSW-193400	318
12.56.3.81 PTSW-193345	318
12.56.3.82 PTSW-193339	319
12.56.3.83 PTSW-193051	319
12.56.3.84 PTSW-192863 / PT-218498	319
12.56.3.85 PTSW-192755	319

12.56.3.86 PTSW-192720 / PT-218467	319
12.56.3.87 PTSW-192635	319
12.56.3.88 PTSW-192456 / PT-218490	319
12.56.3.89 PTSW-192251 / PT-218394	320
12.56.3.90 PTSW-192086 / PT-218465	320
12.56.3.91 PTSW-191875	320
12.56.3.92 PTSW-191446 / PT-218600	320
12.56.3.93 PTSW-191317 / PT-218425	320
12.56.3.94 PTSW-191139	320
12.56.3.95 PTSW-190722	320
12.56.3.96 PTSW-190719	321
12.56.3.97 PTSW-190340	321
12.56.3.98 PTSW-189928 / PT-218456	321
12.56.3.99 PTSW-189738 / PT-218494	321
12.56.3.100 PTSW-189725 / PT-218397	321
12.56.3.101 PTSW-189439 / PT-218427	321
12.56.3.102 PTSW-189279	322
12.56.3.103 PTSW-188836 / PT-218428	322
12.56.3.104 PTSW-188830	322
12.56.3.105 PTSW-188653 / PT-218451	322
12.56.3.106 PTSW-188161	322
12.56.3.107 PTSW-187670	322
12.56.3.108 PTSW-187220 / PT-218584	322
12.56.3.109 PTSW-187216 / PT-218491	323
12.56.3.110 PTSW-187159	323
12.56.3.111 PTSW-187066 / PT-218391	323
12.56.3.112 PTSW-186864	323
12.56.3.113 PTSW-186725	323
12.56.3.114 PTSW-186627	323
12.56.3.115 PTSW-186253	323
12.56.3.116 PTSW-186189	324
12.56.3.117 PTSW-186182	324
12.56.3.118 PTSW-185868 / PT-218439	324
12.56.3.119 PTSW-185867 / PT-218470	324
12.56.3.120 PTSW-185866	324
12.56.3.121 PTSW-185825 / PT-218464	324
12.56.3.122 PTSW-185537	324
12.56.3.123 PTSW-185484	325
12.56.3.124 PTSW-185483	325
12.56.3.125 PTSW-185462	325
12.56.3.126 PTSW-185343	325
12.56.3.127 PTSW-185341	325

12.56.3.128 PTSW-184777 / PT-218483	325
12.56.3.129 PTSW-184770	325
12.56.3.130 PTSW-184682	326
12.56.3.131 PTSW-184642 / PT-218627	326
12.56.3.132 PTSW-184619 / PT-218473 / AAX-600	326
12.56.3.133 PTSW-184541	326
12.56.3.134 PTSW-183902 / PT-218479	326
12.56.3.135 PTSW-183848 / PT-218390	326
12.56.3.136 PTSW-183841	326
12.56.3.137 PTSW-183731	327
12.56.3.138 PTSW-183708	327
12.56.3.139 PTSW-168222	327
12.56.3.140 PTSW-165992	327
12.56.3.141 PTSW-163739	327
12.56.3.142 PTSW-161674	327
12.56.3.143 PTSW-160778	328
12.56.3.144 PTSW-160620	328
12.56.3.145 PTSW-159702	328
12.56.3.146 PTSW-159700	328
12.56.3.147 PTSW-159524	328
12.56.3.148 PTSW-158119	328
12.56.3.149 PTSW-157745	328
12.56.3.150 PTSW-157518	329
12.56.3.151 PTSW-157012	329
12.56.3.152 PTSW-156310	329
12.56.3.153 PTSW-156286	329
12.56.3.154 PTSW-156216	329
12.56.3.155 PTSW-156195	329
12.56.3.156 PTSW-156035	329
12.56.3.157 PTSW-155300 / PT-218458	330
12.56.3.158 PTSW-155177	330
12.56.3.159 PTSW-154361	330
12.56.3.160 PTSW-153140	330
12.56.3.161 PTSW-150047	330
12.56.3.162 PTSW-149880	330
12.56.3.163 PTSW-149819	331
12.56.3.164 PTSW-135536 / PT-218412	331
12.56.3.165 PTSW-3020 / PT-218463	331
12.56.3.166 AAX-686	331
12.56.3.167 AAX-583 / PTSW-157743	331
12.56.4 Known Issues in Venue Live Sound Systems	331
12.56.4.1 VSW-13857	331

12.56.4.2 VSW-13292	332
12.56.4.3 Other Known Issues	332
12.56.5 Known Issues in Media Composer	332
12.56.5.1 MCDEV-2904	332
12.56.6 Known Issues in Control Surfaces	332
12.56.6.1 PT-285383	332
12.56.6.2 PT-226228	333
12.56.6.3 PT-226227	333
12.56.6.4 GWSW-16656	333
12.56.6.5 GWSW-8470	333
12.56.6.6 GWSW-6694	333
12.56.7 Known Issues in Other Software	333
12.56.7.1 XPACE-23	333
12.56.8 Known Issues in AAX Tools	334
12.57 Change Log	334
12.57.1 Change Log	334
12.57.1.1 AAX SDK 2.6.1	334
12.57.1.2 AAX SDK 2.6.0	335
12.57.1.3 AAX SDK 2.5.1	335
12.57.1.4 AAX SDK 2.5.0	336
12.57.1.5 AAX SDK 2.4.1	336
12.57.1.6 AAX SDK 2.4.0	337
12.57.1.7 AAX SDK 2.3.2	338
12.57.1.8 AAX SDK 2.3.1	339
12.57.1.9 AAX SDK 2.3.0	340
12.57.1.10 AAX SDK 2.2.2	342
12.57.1.11 AAX SDK 2.2.1	343
12.57.1.12 AAX SDK 2.2.0	344
12.57.1.13 AAX SDK 2.1.1	346
12.57.1.14 AAX SDK 2.1.0	347
12.57.1.15 AAX SDK 2.0.1	348
12.57.1.16 AAX SDK 2.0.0	348
12.57.1.17 AAX SDK 1.5.0	349
12.57.1.18 AAX SDK 1.0.6	349
12.57.1.19 AAX SDK 1.0.5	349
12.57.1.20 AAX SDK 1.0.4	350
12.57.1.21 AAX SDK 1.0.3	351
12.57.1.22 AAX SDK 1.0.2	352
12.58 Example Plug-Ins	352
12.58.1 SDK Example plug-ins	352
12.58.1.1 Basic examples	353
12.58.1.2 Feature examples	353

12.58.1.3 Deprecated Examples	355
12.59 VENUE Guide	355
12.59.1 Contents	355
12.59.2 About this document	356
12.59.3 Overview of VENUE	356
12.59.4 VENUE systems	356
12.59.4.1 VENUE S6L	356
12.59.4.2 VENUE S3L-X	357
12.59.5 Host environment	357
12.59.5.1 Audio engine	357
12.59.5.2 Available DSP resources	358
12.59.5.3 Operating system	358
12.59.5.4 Display	358
12.59.5.5 Page tables	359
12.59.5.6 Network communications	359
12.59.5.7 Host environment summary	359
12.59.6 AAX feature support and compatibility	360
12.59.6.1 Processing configurations	360
12.59.6.2 Presets and automation	360
12.59.6.3 Unsupported features	361
12.59.7 VENUE Plug-in installer specification	361
12.59.7.1 Overview	362
12.59.7.2 Directory structure	362
12.59.7.3 Optional installer files	363
12.59.7.4 Using a VENUE plug-in installer	365
12.59.8 Additional plug-in guidelines	365
12.59.8.1 General Reliability and Fault Tolerance	365
12.59.8.2 Plug-In Dialogs	365
12.59.8.3 Online Help	365
12.59.9 System details	366
12.59.9.1 External dependencies	366
12.59.9.2 Environment variables	366
12.59.9.3 Plug-in file locations	367
12.59.9.4 Installation process	368
12.59.10 Additional Information	370
12.59.10.1 Metering	370
13 Namespace Documentation	371
13.1 AAX Namespace Reference	371
13.1.1 Enumeration Type Documentation	375
13.1.1.1 EStatusNibble	375
13.1.1.2 EStatusByte	375

13.1.1.3 EChannelModeData	376
13.1.1.4 ESpecialData	376
13.1.1.5 ESampleRates	377
13.1.2 Function Documentation	377
13.1.2.1 AsString() [1/3]	377
13.1.2.2 AsString() [2/3]	377
13.1.2.3 AsString() [3/3]	377
13.1.2.4 IsNoteOn()	378
13.1.2.5 IsNoteOff()	378
13.1.2.6 IsAllNotesOff()	378
13.1.2.7 IsAccentedClick()	379
13.1.2.8 IsUnaccentedClick()	379
13.1.2.9 IsClick()	380
13.1.2.10 PageTableParameterMappingsAreEqual()	380
13.1.2.11 PageTableParameterNameVariationsAreEqual()	381
13.1.2.12 PageTablesAreEqual()	381
13.1.2.13 CopyPageTable()	382
13.1.2.14 FindParameterMappingsInPageTable()	382
13.1.2.15 ClearMappedParameterByID()	383
13.1.2.16 GetCStringOfLength()	383
13.1.2.17 Caseless_strcmp()	383
13.1.2.18 Binary2String()	384
13.1.2.19 String2Binary()	384
13.1.2.20 IsASCII()	385
13.1.2.21 IsFourCharASCII()	385
13.1.2.22 AsStringFourChar()	386
13.1.2.23 AsStringPropertyValue()	386
13.1.2.24 AsStringInt32()	387
13.1.2.25 AsStringUInt32()	387
13.1.2.26 AsStringIDTriad()	387
13.1.2.27 AsStringStemFormat()	388
13.1.2.28 AsStringStemChannel()	388
13.1.2.29 AsStringResult()	388
13.1.2.30 SafeLog()	389
13.1.2.31 SafeLogf()	390
13.1.2.32 IsParameterIDEqual()	390
13.1.2.33 IsEffectIDEqual()	390
13.1.2.34 IsAvidNotification()	390
13.1.2.35 alignFree()	391
13.1.2.36 alignMalloc()	391
13.1.2.37 DeDenormal() [1/2]	391
13.1.2.38 DeDenormal() [2/2]	391

13.1.2.39 DeDenormalFine()	391
13.1.2.40 FilterDenormals()	391
13.1.2.41 ClampToZero()	392
13.1.2.42 ZeroMemorySW()	392
13.1.2.43 ZeroMemoryDW()	392
13.1.2.44 Fill() [1/3]	393
13.1.2.45 Fill() [2/3]	393
13.1.2.46 Fill() [3/3]	394
13.1.2.47 fabs() [1/2]	394
13.1.2.48 fabs() [2/2]	394
13.1.2.49 fabsf()	395
13.1.2.50 AbsMax()	395
13.1.2.51 MinMax()	395
13.1.2.52 Max()	396
13.1.2.53 Min()	396
13.1.2.54 Sign()	396
13.1.2.55 PolyEval()	396
13.1.2.56 CeilLog2()	396
13.1.2.57 SinCosMix()	396
13.1.2.58 FastRound2Int32() [1/2]	396
13.1.2.59 FastRound2Int32() [2/2]	397
13.1.2.60 FastRndDbf2Int32()	398
13.1.2.61 FastTrunc2Int32() [1/2]	398
13.1.2.62 FastTrunc2Int32() [2/2]	399
13.1.2.63 FastRound2Int64()	399
13.1.2.64 GetInt32RPDF()	400
13.1.2.65 GetFastInt32RPDF()	400
13.1.2.66 GetRPDFWithAmplitudeOneHalf()	401
13.1.2.67 GetRPDFWithAmplitudeOne()	401
13.1.2.68 GetFastRPDFWithAmplitudeOne()	402
13.1.2.69 GetTPDFWithAmplitudeOne()	402
13.1.3 Variable Documentation	402
13.1.3.1 cBigEndian	402
13.1.3.2 cLittleEndian	403
13.1.3.3 cPi	403
13.1.3.4 cTwoPi	403
13.1.3.5 cHalfPi	403
13.1.3.6 cQuarterPi	403
13.1.3.7 cRootTwo	403
13.1.3.8 cOneOverRootTwo	403
13.1.3.9 cPos3dB	404
13.1.3.10 cNeg3dB	404

13.1.3.11 cPos6dB	404
13.1.3.12 cNeg6dB	404
13.1.3.13 cNormalizeLongToAmplitudeOneHalf	404
13.1.3.14 cNormalizeLongToAmplitudeOne	404
13.1.3.15 cMilli	404
13.1.3.16 cMicro	405
13.1.3.17 cNano	405
13.1.3.18 cPico	405
13.1.3.19 cKilo	405
13.1.3.20 cMega	405
13.1.3.21 cGiga	405
13.1.3.22 cDenormalAvoidanceOffset	405
13.1.3.23 cFloatDenormalAvoidanceOffset	406
13.1.3.24 kPowExtent	406
13.1.3.25 kPowTableSize	406
13.1.3.26 cSeedDivisor	406
13.1.3.27 cInitialSeedValue	406
13.2 AAX::Exception Namespace Reference	406
13.2.1 Description	406
13.3 AAX::internal Namespace Reference	407
13.3.1 Function Documentation	407
13.3.1.1 ToHexadecimal()	407
13.4 AAX_ChunkDataParserDefs Namespace Reference	407
13.4.1 Description	407
13.4.2 Variable Documentation	408
13.4.2.1 FLOAT_TYPE	408
13.4.2.2 FLOAT_STRING_IDENTIFIER	408
13.4.2.3 LONG_TYPE	408
13.4.2.4 LONG_STRING_IDENTIFIER	408
13.4.2.5 DOUBLE_TYPE	408
13.4.2.6 DOUBLE_STRING_IDENTIFIER	408
13.4.2.7 DOUBLE_TYPE_SIZE	408
13.4.2.8 DOUBLE_TYPE_INCR	409
13.4.2.9 SHORT_TYPE	409
13.4.2.10 SHORT_STRING_IDENTIFIER	409
13.4.2.11 SHORT_TYPE_SIZE	409
13.4.2.12 SHORT_TYPE_INCR	409
13.4.2.13 STRING_TYPE	409
13.4.2.14 STRING_STRING_IDENTIFIER	409
13.4.2.15 MAX_STRINGDATA_LENGTH	409
13.4.2.16 DEFAULT32BIT_TYPE_SIZE	410
13.4.2.17 DEFAULT32BIT_TYPE_INCR	410

13.4.2.18 STRING_IDENTIFIER_SIZE	410
13.4.2.19 NAME_NOT_FOUND	410
13.4.2.20 MAX_NAME_LENGTH	410
13.4.2.21 BUILD_DATA_FAILED	410
13.4.2.22 HEADER_SIZE	410
13.4.2.23 VERSION_ID_1	410
14 Class Documentation	411
14.1 _acfUID Struct Reference	411
14.1.1 Member Data Documentation	411
14.1.1.1 Data1	411
14.1.1.2 Data2	411
14.1.1.3 Data3	411
14.1.1.4 Data4	412
14.2 AAX_AggregateResult Class Reference	412
14.2.1 Description	412
14.2.2 Constructor & Destructor Documentation	413
14.2.2.1 AAX_AggregateResult()	413
14.2.2.2 ~AAX_AggregateResult()	413
14.2.3 Member Function Documentation	413
14.2.3.1 operator=()	413
14.2.3.2 operator AAX_Result()	414
14.2.3.3 Check()	414
14.2.3.4 Clear()	415
14.2.3.5 LastFailure()	415
14.2.3.6 NumFailed()	415
14.2.3.7 NumSucceeded()	416
14.2.3.8 NumAttempted()	416
14.3 AAX_CArrayDataBuffer< D > Class Template Reference	416
14.3.1 Description	417
14.3.2 Constructor & Destructor Documentation	418
14.3.2.1 AAX_CArrayDataBuffer() [1/4]	418
14.3.2.2 AAX_CArrayDataBuffer() [2/4]	418
14.3.2.3 AAX_CArrayDataBuffer() [3/4]	418
14.3.2.4 AAX_CArrayDataBuffer() [4/4]	419
14.3.2.5 ~AAX_CArrayDataBuffer()	419
14.3.3 Member Function Documentation	419
14.3.3.1 operator=() [1/2]	419
14.3.3.2 operator=() [2/2]	419
14.3.3.3 Type()	419
14.3.3.4 Size()	420
14.3.3.5 Data()	420

14.4 AAX_CArrayDataBufferOfType< T, D > Class Template Reference	420
14.4.1 Description	421
14.4.2 Constructor & Destructor Documentation	422
14.4.2.1 AAX_CArrayDataBufferOfType() [1/4]	422
14.4.2.2 AAX_CArrayDataBufferOfType() [2/4]	422
14.4.2.3 AAX_CArrayDataBufferOfType() [3/4]	422
14.4.2.4 AAX_CArrayDataBufferOfType() [4/4]	423
14.4.2.5 ~AAX_CArrayDataBufferOfType()	423
14.4.3 Member Function Documentation	423
14.4.3.1 operator=() [1/2]	423
14.4.3.2 operator=() [2/2]	423
14.4.3.3 Type()	423
14.4.3.4 Size()	424
14.4.3.5 Data()	424
14.5 AAX_CAtomicQueue< T, S > Class Template Reference	424
14.5.1 Description	425
14.5.2 Member Typedef Documentation	426
14.5.2.1 template_type	427
14.5.2.2 value_type	427
14.5.3 Constructor & Destructor Documentation	427
14.5.3.1 ~AAX_CAtomicQueue()	427
14.5.3.2 AAX_CAtomicQueue()	427
14.5.4 Member Function Documentation	427
14.5.4.1 Clear()	427
14.5.4.2 Push()	428
14.5.4.3 Pop()	428
14.5.4.4 Peek()	429
14.5.5 Member Data Documentation	429
14.5.5.1 template_size	429
14.6 AAX_CAutoreleasePool Class Reference	429
14.6.1 Constructor & Destructor Documentation	429
14.6.1.1 AAX_CAutoreleasePool()	430
14.6.1.2 ~AAX_CAutoreleasePool()	430
14.7 AAX_CBinaryDisplayDelegate< T > Class Template Reference	430
14.7.1 Description	431
14.7.2 Constructor & Destructor Documentation	432
14.7.2.1 AAX_CBinaryDisplayDelegate() [1/2]	432
14.7.2.2 AAX_CBinaryDisplayDelegate() [2/2]	432
14.7.3 Member Function Documentation	432
14.7.3.1 Clone()	433
14.7.3.2 ValueToString() [1/2]	433
14.7.3.3 ValueToString() [2/2]	433

14.7.3.4 StringToValue()	434
14.7.3.5 AddShortenedStrings()	434
14.8 AAX_CBinaryTaperDelegate< T > Class Template Reference	435
14.8.1 Description	435
14.8.2 Constructor & Destructor Documentation	436
14.8.2.1 AAX_CBinaryTaperDelegate()	436
14.8.3 Member Function Documentation	437
14.8.3.1 Clone()	437
14.8.3.2 GetMaximumValue()	437
14.8.3.3 GetMinimumValue()	437
14.8.3.4 ConstrainRealValue()	437
14.8.3.5 NormalizedToReal()	438
14.8.3.6 RealToNormalized()	438
14.9 AAX_CChunkDataParser Class Reference	439
14.9.1 Description	439
14.9.2 Constructor & Destructor Documentation	441
14.9.2.1 AAX_CChunkDataParser()	441
14.9.2.2 ~AAX_CChunkDataParser()	441
14.9.3 Member Function Documentation	442
14.9.3.1 AddFloat()	442
14.9.3.2 AddDouble()	442
14.9.3.3 AddInt32()	442
14.9.3.4 AddInt16()	442
14.9.3.5 AddString()	443
14.9.3.6 FindFloat()	443
14.9.3.7 FindDouble()	443
14.9.3.8 FindInt32()	443
14.9.3.9 FindInt16()	443
14.9.3.10 FindString()	444
14.9.3.11 ReplaceDouble()	444
14.9.3.12 GetChunkData()	444
14.9.3.13 GetChunkDataSize()	444
14.9.3.14 GetChunkVersion()	444
14.9.3.15 IsEmpty()	444
14.9.3.16 Clear()	445
14.9.3.17 LoadChunk()	445
14.9.3.18 WordAlign() [1/2]	445
14.9.3.19 WordAlign() [2/2]	445
14.9.3.20 FindName()	445
14.9.4 Member Data Documentation	445
14.9.4.1 mLastFoundIndex	446
14.9.4.2 mChunkData	446

14.9.4.3 mChunkVersion	446
14.9.4.4 mDataValues	446
14.10 AAX_CDecibelDisplayDelegateDecorator< T > Class Template Reference	446
14.10.1 Description	447
14.10.2 Constructor & Destructor Documentation	449
14.10.2.1 AAX_CDecibelDisplayDelegateDecorator()	449
14.10.3 Member Function Documentation	449
14.10.3.1 Clone()	449
14.10.3.2 ValueToString() [1/2]	449
14.10.3.3 ValueToString() [2/2]	450
14.10.3.4 StringToValue()	451
14.11 AAX_CEffectDirectData Class Reference	452
14.11.1 Description	453
14.11.2 Constructor & Destructor Documentation	455
14.11.2.1 AAX_CEffectDirectData()	455
14.11.2.2 ~AAX_CEffectDirectData()	455
14.11.3 Member Function Documentation	455
14.11.3.1 Initialize()	455
14.11.3.2 Uninitialize()	456
14.11.3.3 TimerWakeup()	456
14.11.3.4 NotificationReceived()	456
14.11.3.5 Controller()	457
14.11.3.6 EffectParameters()	457
14.11.3.7 Initialize_PrivateDataAccess()	457
14.11.3.8 TimerWakeup_PrivateDataAccess()	457
14.12 AAX_CEffectGUI Class Reference	458
14.12.1 Description	459
14.12.2 Constructor & Destructor Documentation	462
14.12.2.1 AAX_CEffectGUI()	462
14.12.2.2 ~AAX_CEffectGUI()	462
14.12.3 Member Function Documentation	462
14.12.3.1 Initialize()	462
14.12.3.2 Uninitialize()	462
14.12.3.3 NotificationReceived()	463
14.12.3.4 SetViewContainer()	463
14.12.3.5 GetViewSize()	464
14.12.3.6 Draw()	464
14.12.3.7 TimerWakeup()	465
14.12.3.8 ParameterUpdated()	465
14.12.3.9 GetCustomLabel()	465
14.12.3.10 SetControlHighlightInfo()	466
14.12.3.11 CreateViewContents()	466

14.12.3.12 CreateViewContainer()	466
14.12.3.13 DeleteViewContainer()	467
14.12.3.14 UpdateAllParameters()	467
14.12.3.15 GetController() [1/2]	467
14.12.3.16 GetController() [2/2]	467
14.12.3.17 GetEffectParameters() [1/2]	467
14.12.3.18 GetEffectParameters() [2/2]	468
14.12.3.19 GetViewContainer() [1/2]	468
14.12.3.20 GetViewContainer() [2/2]	468
14.12.3.21 Transport() [1/2]	468
14.12.3.22 Transport() [2/2]	468
14.12.3.23 GetViewContainerType()	468
14.12.3.24 GetViewContainerPtr()	469
14.13 AAX_CEffectParameters Class Reference	469
14.13.1 Description	470
14.13.2 Related classes	470
14.13.3 Constructor & Destructor Documentation	478
14.13.3.1 AAX_CEffectParameters()	478
14.13.3.2 ~AAX_CEffectParameters()	478
14.13.4 Member Function Documentation	478
14.13.4.1 operator=()	478
14.13.4.2 Initialize()	478
14.13.4.3 Uninitialize()	479
14.13.4.4 NotificationReceived()	479
14.13.4.5 GetNumberOfParameters()	480
14.13.4.6 GetMasterBypassParameter()	480
14.13.4.7 GetParameterIsAutomatable()	480
14.13.4.8 GetParameterNumberOfSteps()	481
14.13.4.9 GetParameterName()	481
14.13.4.10 GetParameterNameOfLength()	481
14.13.4.11 GetParameterDefaultNormalizedValue()	482
14.13.4.12 SetParameterDefaultNormalizedValue()	482
14.13.4.13 GetParameterType()	483
14.13.4.14 GetParameterOrientation()	483
14.13.4.15 GetParameter()	484
14.13.4.16 GetParameterIndex()	484
14.13.4.17 GetParameterIDFromIndex()	485
14.13.4.18 GetParameterValueInfo()	485
14.13.4.19 GetParameterValueFromString()	485
14.13.4.20 GetParameterStringFromValue()	486
14.13.4.21 GetParameterValueString()	487
14.13.4.22 GetParameterNormalizedValue()	487

14.13.4.23 SetParameterNormalizedValue()	487
14.13.4.24 SetParameterNormalizedRelative()	488
14.13.4.25 TouchParameter()	488
14.13.4.26 ReleaseParameter()	489
14.13.4.27 UpdateParameterTouch()	489
14.13.4.28 UpdateParameterNormalizedValue()	490
14.13.4.29 UpdateParameterNormalizedRelative()	490
14.13.4.30 GenerateCoefficients()	491
14.13.4.31 ResetFieldData()	492
14.13.4.32 GetNumberOfChunks()	493
14.13.4.33 GetChunkIDFromIndex()	493
14.13.4.34 GetChunkSize()	493
14.13.4.35 GetChunk()	494
14.13.4.36 SetChunk()	495
14.13.4.37 CompareActiveChunk()	495
14.13.4.38 GetNumberOfChanges()	496
14.13.4.39 TimerWakeup()	496
14.13.4.40 GetCurveData()	497
14.13.4.41 GetCurveDataMeterIds()	498
14.13.4.42 GetCurveDataDisplayRange()	498
14.13.4.43 UpdatePageTable() [1/2]	499
14.13.4.44 GetCustomData()	500
14.13.4.45 SetCustomData()	500
14.13.4.46 DoMIDITransfers()	500
14.13.4.47 UpdateMIDINodes()	501
14.13.4.48 UpdateControlMIDINodes()	501
14.13.4.49 RenderAudio_Hybrid()	502
14.13.4.50 Controller() [1/2]	502
14.13.4.51 Controller() [2/2]	502
14.13.4.52 Transport() [1/2]	502
14.13.4.53 Transport() [2/2]	503
14.13.4.54 AutomationDelegate() [1/2]	503
14.13.4.55 AutomationDelegate() [2/2]	503
14.13.4.56 SetTaperDelegate()	503
14.13.4.57 SetDisplayDelegate()	503
14.13.4.58 IsParameterTouched()	503
14.13.4.59 IsParameterLinkReady()	504
14.13.4.60 EffectInit()	504
14.13.4.61 UpdatePageTable() [2/2]	504
14.13.4.62 FilterParameterIDOnSave()	505
14.13.4.63 BuildChunkData()	505
14.13.5 Member Data Documentation	505

14.13.5.1 mNumPluginChanges	505
14.13.5.2 mChunkSize	505
14.13.5.3 mChunkParser	505
14.13.5.4 mNumChunkedParameters	506
14.13.5.5 mPacketDispatcher	506
14.13.5.6 mParameterManager	506
14.13.5.7 mFilteredParameters	506
14.14 AAX_CheckedResult Class Reference	506
14.14.1 Description	506
14.14.2 Member Typedef Documentation	508
14.14.2.1 Exception	508
14.14.3 Constructor & Destructor Documentation	509
14.14.3.1 ~AAX_CheckedResult()	509
14.14.3.2 AAX_CheckedResult() [1/2]	509
14.14.3.3 AAX_CheckedResult() [2/2]	509
14.14.4 Member Function Documentation	509
14.14.4.1 AddAcceptedResult()	509
14.14.4.2 ResetAcceptedResults()	509
14.14.4.3 operator=()	510
14.14.4.4 operator" =()	510
14.14.4.5 operator AAX_Result()	511
14.14.4.6 Clear()	511
14.14.4.7 LastError()	511
14.15 AAX_CHostProcessor Class Reference	511
14.15.1 Description	513
14.15.2 Constructor & Destructor Documentation	516
14.15.2.1 AAX_CHostProcessor()	516
14.15.2.2 ~AAX_CHostProcessor()	516
14.15.3 Member Function Documentation	516
14.15.3.1 Initialize()	516
14.15.3.2 Uninitialize()	516
14.15.3.3 InitOutputBounds()	517
14.15.3.4 SetLocation()	518
14.15.3.5 RenderAudio()	518
14.15.3.6 PreRender()	519
14.15.3.7 PostRender()	519
14.15.3.8 AnalyzeAudio()	520
14.15.3.9 PreAnalyze()	520
14.15.3.10 PostAnalyze()	521
14.15.3.11 GetClipNameSuffix()	521
14.15.3.12 GetEffectParameters() [1/2]	521
14.15.3.13 GetEffectParameters() [2/2]	521

14.15.3.14	GetHostProcessorDelegate() [1/2]	522
14.15.3.15	GetHostProcessorDelegate() [2/2]	522
14.15.3.16	GetLocation()	522
14.15.3.17	GetInputRange()	522
14.15.3.18	GetOutputRange()	522
14.15.3.19	GetSrcStart()	522
14.15.3.20	GetSrcEnd()	523
14.15.3.21	GetDstStart()	523
14.15.3.22	GetDstEnd()	523
14.15.3.23	TranslateOutputBounds()	523
14.15.3.24	GetAudio()	524
14.15.3.25	GetSideChainInputNum()	524
14.15.3.26	Controller() [1/2]	524
14.15.3.27	Controller() [2/2]	524
14.15.3.28	HostProcessorDelegate() [1/2]	525
14.15.3.29	HostProcessorDelegate() [2/2]	525
14.15.3.30	EffectParameters() [1/2]	525
14.15.3.31	EffectParameters() [2/2]	525
14.16	AAX_CHostServices Class Reference	525
14.16.1	Description	525
14.16.2	Member Function Documentation	526
14.16.2.1	Set()	526
14.16.2.2	HandleAssertFailure()	526
14.16.2.3	Trace()	526
14.16.2.4	StackTrace()	527
14.17	AAX_CLinearTaperDelegate< T, RealPrecision > Class Template Reference	527
14.17.1	Description	528
14.17.2	Constructor & Destructor Documentation	530
14.17.2.1	AAX_CLinearTaperDelegate()	530
14.17.3	Member Function Documentation	530
14.17.3.1	Clone()	530
14.17.3.2	GetMinimumValue()	531
14.17.3.3	GetMaximumValue()	531
14.17.3.4	ConstrainRealValue()	531
14.17.3.5	NormalizedToReal()	531
14.17.3.6	RealToNormalized()	532
14.17.3.7	Round()	532
14.18	AAX_CLogTaperDelegate< T, RealPrecision > Class Template Reference	533
14.18.1	Description	533
14.18.2	Constructor & Destructor Documentation	535
14.18.2.1	AAX_CLogTaperDelegate()	535
14.18.3	Member Function Documentation	535

14.18.3.1 Clone()	535
14.18.3.2 GetMinimumValue()	536
14.18.3.3 GetMaximumValue()	536
14.18.3.4 ConstrainRealValue()	536
14.18.3.5 NormalizedToReal()	536
14.18.3.6 RealToNormalized()	537
14.18.3.7 Round()	538
14.19 AAX_CMidiPacket Struct Reference	538
14.19.1 Description	538
14.19.2 Member Data Documentation	539
14.19.2.1 mTimestamp	539
14.19.2.2 mLength	539
14.19.2.3 mData	539
14.19.2.4 mIsImmediate	539
14.20 AAX_CMidiStream Struct Reference	540
14.20.1 Description	540
14.20.2 Member Data Documentation	540
14.20.2.1 mBufferSize	541
14.20.2.2 mBuffer	541
14.21 AAX_CMonolithicParameters Class Reference	541
14.21.1 Description	542
14.21.2 Member Typedef Documentation	549
14.21.2.1 TParamValPair	550
14.21.3 Constructor & Destructor Documentation	550
14.21.3.1 AAX_CMonolithicParameters()	550
14.21.3.2 ~AAX_CMonolithicParameters()	550
14.21.4 Member Function Documentation	550
14.21.4.1 RenderAudio()	550
14.21.4.2 AddSynchronizedParameter()	551
14.21.4.3 UpdateParameterNormalizedValue()	552
14.21.4.4 GenerateCoefficients()	553
14.21.4.5 ResetFieldData()	554
14.21.4.6 TimerWakeup()	555
14.21.4.7 StaticDescribe()	556
14.21.4.8 StaticRenderAudio()	557
14.22 AAX_CMutex Class Reference	558
14.22.1 Description	558
14.22.2 Constructor & Destructor Documentation	559
14.22.2.1 AAX_CMutex()	559
14.22.2.2 ~AAX_CMutex()	559
14.22.3 Member Function Documentation	559
14.22.3.1 Lock()	559

14.22.3.2 Unlock()	560
14.22.3.3 Try_Lock()	560
14.23 AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter > Class Template Reference	560
14.23.1 Description	561
14.23.2 Member Function Documentation	562
14.23.2.1 Clone()	562
14.23.2.2 ValueToString() [1/2]	562
14.23.2.3 ValueToString() [2/2]	563
14.23.2.4 StringToValue()	564
14.24 AAX_Component< aContextType > Class Template Reference	565
14.24.1 Description	565
14.24.2 Member Typedef Documentation	565
14.24.2.1 CProcessProc	566
14.24.2.2 CPacketAllocator	566
14.24.2.3 CInstanceInitProc	566
14.24.2.4 CBackgroundProc	566
14.24.2.5 CInitPrivateDataProc	566
14.25 AAX_CPacket Class Reference	566
14.25.1 Description	567
14.25.2 Constructor & Destructor Documentation	567
14.25.2.1 AAX_CPacket()	567
14.25.2.2 ~AAX_CPacket()	567
14.25.3 Member Function Documentation	567
14.25.3.1 GetPtr() [1/2]	567
14.25.3.2 SetDirty()	568
14.25.3.3 IsDirty()	568
14.25.3.4 GetID()	568
14.25.3.5 GetSize()	568
14.25.3.6 GetPtr() [2/2]	568
14.26 AAX_CPacketDispatcher Class Reference	568
14.26.1 Description	568
14.26.2 Constructor & Destructor Documentation	569
14.26.2.1 AAX_CPacketDispatcher()	569
14.26.2.2 ~AAX_CPacketDispatcher()	569
14.26.3 Member Function Documentation	569
14.26.3.1 Initialize()	569
14.26.3.2 RegisterPacket() [1/3]	570
14.26.3.3 RegisterPacket() [2/3]	570
14.26.3.4 RegisterPacket() [3/3]	571
14.26.3.5 SetDirty()	571
14.26.3.6 Dispatch()	571
14.26.3.7 GenerateSingleValuePacket()	571

14.27 AAX_CPacketHandler< TWorker > Class Template Reference	572
14.27.1 Description	573
14.27.2 Constructor & Destructor Documentation	573
14.27.2.1 AAX_CPacketHandler() [1/2]	573
14.27.2.2 AAX_CPacketHandler() [2/2]	573
14.27.3 Member Function Documentation	574
14.27.3.1 Clone()	574
14.27.3.2 Call()	574
14.27.4 Member Data Documentation	574
14.27.4.1 pt2Object	574
14.27.4.2 fpt	574
14.27.4.3 fptEx	575
14.28 AAX_CParameter< T > Class Template Reference	575
14.28.1 Description	576
14.28.2 Member Enumeration Documentation	580
14.28.2.1 Type	580
14.28.2.2 Defaults	581
14.28.3 Constructor & Destructor Documentation	581
14.28.3.1 AAX_CParameter() [1/4]	581
14.28.3.2 AAX_CParameter() [2/4]	582
14.28.3.3 AAX_CParameter() [3/4]	583
14.28.3.4 AAX_CParameter() [4/4]	583
14.28.3.5 ~AAX_CParameter()	584
14.28.4 Member Function Documentation	584
14.28.4.1 AAX_DEFAULT_MOVE_CTOR()	584
14.28.4.2 AAX_DEFAULT_MOVE_OPER()	584
14.28.4.3 AAX_DELETE() [1/3]	585
14.28.4.4 AAX_DELETE() [2/3]	585
14.28.4.5 AAX_DELETE() [3/3]	585
14.28.4.6 CloneValue()	585
14.28.4.7 Identifier()	585
14.28.4.8 SetName()	585
14.28.4.9 Name()	586
14.28.4.10 AddShortenedName()	586
14.28.4.11 ShortenedName()	586
14.28.4.12 ClearShortenedNames()	587
14.28.4.13 SetNormalizedDefaultValue()	587
14.28.4.14 GetNormalizedDefaultValue()	587
14.28.4.15 SetToDefaultValue()	588
14.28.4.16 SetNormalizedValue()	588
14.28.4.17 GetNormalizedValue()	588
14.28.4.18 SetNumberOfSteps()	589

14.28.4.19 GetNumberOfSteps()	589
14.28.4.20 GetStepValue()	589
14.28.4.21 GetNormalizedValueFromStep()	590
14.28.4.22 GetStepValueFromNormalizedValue()	590
14.28.4.23 SetStepValue()	590
14.28.4.24 SetType()	590
14.28.4.25 GetType()	591
14.28.4.26 SetOrientation()	591
14.28.4.27 GetOrientation()	591
14.28.4.28 SetTaperDelegate()	592
14.28.4.29 SetDisplayDelegate()	592
14.28.4.30 GetValueString() [1/2]	593
14.28.4.31 GetValueString() [2/2]	593
14.28.4.32 GetNormalizedValueFromBool() [1/2]	594
14.28.4.33 GetNormalizedValueFromInt32() [1/2]	594
14.28.4.34 GetNormalizedValueFromFloat() [1/2]	595
14.28.4.35 GetNormalizedValueFromDouble() [1/2]	595
14.28.4.36 GetNormalizedValueFromString()	596
14.28.4.37 GetBoolFromNormalizedValue() [1/2]	596
14.28.4.38 GetInt32FromNormalizedValue() [1/2]	597
14.28.4.39 GetFloatFromNormalizedValue() [1/2]	597
14.28.4.40 GetDoubleFromNormalizedValue() [1/2]	597
14.28.4.41 GetStringFromNormalizedValue() [1/2]	598
14.28.4.42 GetStringFromNormalizedValue() [2/2]	598
14.28.4.43 SetValueFromString()	599
14.28.4.44 SetAutomationDelegate()	599
14.28.4.45 Automatable()	600
14.28.4.46 Touch()	600
14.28.4.47 Release()	601
14.28.4.48 GetValueAsBool()	601
14.28.4.49 GetValueAsInt32()	601
14.28.4.50 GetValueAsFloat()	602
14.28.4.51 GetValueAsDouble()	602
14.28.4.52 GetValueAsString() [1/2]	603
14.28.4.53 SetValueWithBool() [1/2]	603
14.28.4.54 SetValueWithInt32() [1/2]	603
14.28.4.55 SetValueWithFloat() [1/2]	604
14.28.4.56 SetValueWithDouble() [1/2]	604
14.28.4.57 SetValueWithString() [1/2]	605
14.28.4.58 UpdateNormalizedValue()	605
14.28.4.59 SetValue()	606
14.28.4.60 GetValue()	606

14.28.4.61 SetDefaultValue()	606
14.28.4.62 GetDefaultValue()	606
14.28.4.63 TaperDelegate()	607
14.28.4.64 DisplayDelegate()	607
14.28.4.65 GetValueAsString() [2/2]	607
14.28.4.66 SetValueWithBool() [2/2]	607
14.28.4.67 SetValueWithInt32() [2/2]	608
14.28.4.68 SetValueWithFloat() [2/2]	608
14.28.4.69 SetValueWithDouble() [2/2]	609
14.28.4.70 SetValueWithString() [2/2]	609
14.28.4.71 GetNormalizedValueFromBool() [2/2]	609
14.28.4.72 GetNormalizedValueFromInt32() [2/2]	610
14.28.4.73 GetNormalizedValueFromFloat() [2/2]	610
14.28.4.74 GetNormalizedValueFromDouble() [2/2]	611
14.28.4.75 GetBoolFromNormalizedValue() [2/2]	611
14.28.4.76 GetInt32FromNormalizedValue() [2/2]	612
14.28.4.77 GetFloatFromNormalizedValue() [2/2]	612
14.28.4.78 GetDoubleFromNormalizedValue() [2/2]	613
14.28.5 Member Data Documentation	613
14.28.5.1 mName	613
14.28.5.2 mAutomatable	613
14.28.5.3 mNumSteps	613
14.28.5.4 mControlType	614
14.28.5.5 mOrientation	614
14.28.5.6 mTaperDelegate	614
14.28.5.7 mDisplayDelegate	614
14.28.5.8 mAutomationDelegate	614
14.28.5.9 mNeedNotify	614
14.28.5.10 mValue	614
14.28.5.11 mDefaultValue	615
14.29 AAX_CParameterManager Class Reference	615
14.29.1 Description	615
14.29.2 Constructor & Destructor Documentation	616
14.29.2.1 AAX_CParameterManager()	616
14.29.2.2 ~AAX_CParameterManager()	616
14.29.3 Member Function Documentation	616
14.29.3.1 Initialize()	616
14.29.3.2 NumParameters()	617
14.29.3.3 RemoveParameterByID()	617
14.29.3.4 RemoveAllParameters()	617
14.29.3.5 GetParameterByID() [1/2]	617
14.29.3.6 GetParameterByID() [2/2]	618

14.29.3.7 GetParameterByName() [1/2]	618
14.29.3.8 GetParameterByName() [2/2]	619
14.29.3.9 GetParameter() [1/2]	619
14.29.3.10 GetParameter() [2/2]	619
14.29.3.11 GetParameterIndex()	620
14.29.3.12 AddParameter()	620
14.29.3.13 RemoveParameter()	620
14.29.4 Member Data Documentation	620
14.29.4.1 mAutomationDelegate	621
14.29.4.2 mParameters	621
14.29.4.3 mParametersMap	621
14.30 AAX_CParameterValue< T > Class Template Reference	621
14.30.1 Description	622
14.30.2 Member Enumeration Documentation	623
14.30.2.1 Defaults	623
14.30.3 Constructor & Destructor Documentation	623
14.30.3.1 AAX_CParameterValue() [1/3]	623
14.30.3.2 AAX_CParameterValue() [2/3]	624
14.30.3.3 AAX_CParameterValue() [3/3]	624
14.30.4 Member Function Documentation	624
14.30.4.1 AAX_DEFAULT_DTOR_OVERRIDE()	624
14.30.4.2 AAX_DEFAULT_MOVE_CTOR()	625
14.30.4.3 AAX_DEFAULT_MOVE_OPER()	625
14.30.4.4 AAX_DELETE()	625
14.30.4.5 Get()	625
14.30.4.6 Set()	625
14.30.4.7 Clone()	626
14.30.4.8 Identifier()	626
14.30.4.9 GetValueAsBool() [1/2]	626
14.30.4.10 GetValueAsInt32() [1/2]	627
14.30.4.11 GetValueAsFloat() [1/2]	627
14.30.4.12 GetValueAsDouble() [1/2]	627
14.30.4.13 GetValueAsString() [1/2]	628
14.30.4.14 GetValueAsBool() [2/2]	628
14.30.4.15 GetValueAsInt32() [2/2]	629
14.30.4.16 GetValueAsFloat() [2/2]	629
14.30.4.17 GetValueAsDouble() [2/2]	630
14.30.4.18 GetValueAsString() [2/2]	630
14.31 AAX_CPercentDisplayDelegateDecorator< T > Class Template Reference	630
14.31.1 Description	631
14.31.2 Constructor & Destructor Documentation	633
14.31.2.1 AAX_CPercentDisplayDelegateDecorator()	633

14.31.3 Member Function Documentation	633
14.31.3.1 Clone()	633
14.31.3.2 ValueToString() [1/2]	633
14.31.3.3 ValueToString() [2/2]	634
14.31.3.4 StringToValue()	635
14.32 AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision > Class Template Reference	636
14.32.1 Description	637
14.32.2 Constructor & Destructor Documentation	638
14.32.2.1 AAX_CPieceWiseLinearTaperDelegate() [1/2]	638
14.32.2.2 AAX_CPieceWiseLinearTaperDelegate() [2/2]	638
14.32.2.3 ~AAX_CPieceWiseLinearTaperDelegate()	638
14.32.3 Member Function Documentation	639
14.32.3.1 Clone()	639
14.32.3.2 GetMinimumValue()	639
14.32.3.3 GetMaximumValue()	639
14.32.3.4 ConstrainRealValue()	639
14.32.3.5 NormalizedToReal()	640
14.32.3.6 RealToNormalized()	640
14.32.3.7 Round()	641
14.33 AAX_CRangeTaperDelegate< T, RealPrecision > Class Template Reference	641
14.33.1 Description	642
14.33.2 Constructor & Destructor Documentation	643
14.33.2.1 AAX_CRangeTaperDelegate() [1/2]	644
14.33.2.2 AAX_CRangeTaperDelegate() [2/2]	644
14.33.3 Member Function Documentation	644
14.33.3.1 operator=()	644
14.33.3.2 Clone()	645
14.33.3.3 GetMinimumValue()	645
14.33.3.4 GetMaximumValue()	645
14.33.3.5 ConstrainRealValue()	645
14.33.3.6 NormalizedToReal()	646
14.33.3.7 RealToNormalized()	646
14.33.3.8 Round()	647
14.33.3.9 SmartRound()	647
14.34 AAX_CSessionDocumentClient Class Reference	647
14.34.1 Description	648
14.34.2 Constructor & Destructor Documentation	650
14.34.2.1 AAX_CSessionDocumentClient()	650
14.34.2.2 ~AAX_CSessionDocumentClient()	650
14.34.3 Member Function Documentation	650
14.34.3.1 Initialize()	650
14.34.3.2 Uninitialize()	650

14.34.3.3 SetSessionDocument()	650
14.34.3.4 NotificationReceived()	651
14.34.3.5 SessionDocumentWillChange()	651
14.34.3.6 SessionDocumentChanged()	652
14.34.3.7 GetController() [1/2]	652
14.34.3.8 GetController() [2/2]	652
14.34.3.9 GetEffectParameters() [1/2]	652
14.34.3.10 GetEffectParameters() [2/2]	653
14.34.3.11 GetSessionDocument() [1/2]	653
14.34.3.12 GetSessionDocument() [2/2]	653
14.35 AAX_CStateDisplayDelegate< T > Class Template Reference	653
14.35.1 Description	654
14.35.2 Constructor & Destructor Documentation	655
14.35.2.1 AAX_CStateDisplayDelegate() [1/4]	655
14.35.2.2 AAX_CStateDisplayDelegate() [2/4]	655
14.35.2.3 AAX_CStateDisplayDelegate() [3/4]	656
14.35.2.4 AAX_CStateDisplayDelegate() [4/4]	656
14.35.3 Member Function Documentation	656
14.35.3.1 Clone()	656
14.35.3.2 ValueToString() [1/2]	656
14.35.3.3 ValueToString() [2/2]	657
14.35.3.4 StringToValue()	657
14.35.3.5 AddShortenedStrings()	658
14.35.3.6 Compare()	658
14.36 AAX_CStatelessParameter Class Reference	658
14.36.1 Description	659
14.36.2 Constructor & Destructor Documentation	662
14.36.2.1 AAX_CStatelessParameter() [1/2]	662
14.36.2.2 AAX_CStatelessParameter() [2/2]	662
14.36.3 Member Function Documentation	662
14.36.3.1 AAX_DEFAULT_DTOR_OVERRIDE()	662
14.36.3.2 CloneValue()	662
14.36.3.3 Identifier()	663
14.36.3.4 SetName()	663
14.36.3.5 Name()	664
14.36.3.6 AddShortenedName()	665
14.36.3.7 ShortenedName()	665
14.36.3.8 ClearShortenedNames()	666
14.36.3.9 Automatable()	666
14.36.3.10 SetAutomationDelegate()	666
14.36.3.11 Touch()	667
14.36.3.12 Release()	668

14.36.3.13 SetNormalizedValue()	668
14.36.3.14 GetNormalizedValue()	668
14.36.3.15 SetNormalizedDefaultValue()	669
14.36.3.16 GetNormalizedDefaultValue()	669
14.36.3.17 SetToDefaultValue()	669
14.36.3.18 SetNumberOfSteps()	669
14.36.3.19 GetNumberOfSteps()	670
14.36.3.20 GetStepValue()	670
14.36.3.21 GetNormalizedValueFromStep()	670
14.36.3.22 GetStepValueFromNormalizedValue()	670
14.36.3.23 SetStepValue()	671
14.36.3.24 GetValueString() [1/2]	671
14.36.3.25 GetValueString() [2/2]	671
14.36.3.26 GetNormalizedValueFromBool()	672
14.36.3.27 GetNormalizedValueFromInt32()	673
14.36.3.28 GetNormalizedValueFromFloat()	673
14.36.3.29 GetNormalizedValueFromDouble()	674
14.36.3.30 GetNormalizedValueFromString()	674
14.36.3.31 GetBoolFromNormalizedValue()	675
14.36.3.32 GetInt32FromNormalizedValue()	675
14.36.3.33 GetFloatFromNormalizedValue()	675
14.36.3.34 GetDoubleFromNormalizedValue()	676
14.36.3.35 GetStringFromNormalizedValue() [1/2]	676
14.36.3.36 GetStringFromNormalizedValue() [2/2]	677
14.36.3.37 SetValueFromString()	678
14.36.3.38 GetValueAsBool()	678
14.36.3.39 GetValueAsInt32()	679
14.36.3.40 GetValueAsFloat()	679
14.36.3.41 GetValueAsDouble()	680
14.36.3.42 GetValueAsString()	680
14.36.3.43 SetValueWithBool()	680
14.36.3.44 SetValueWithInt32()	681
14.36.3.45 SetValueWithFloat()	681
14.36.3.46 SetValueWithDouble()	682
14.36.3.47 SetValueWithString()	682
14.36.3.48 SetType()	683
14.36.3.49 GetType()	683
14.36.3.50 SetOrientation()	683
14.36.3.51 GetOrientation()	683
14.36.3.52 SetTaperDelegate()	684
14.36.3.53 SetDisplayDelegate()	684
14.36.3.54 UpdateNormalizedValue()	684

14.36.4 Member Data Documentation	685
14.36.4.1 mNames	685
14.36.4.2 mID	685
14.36.4.3 mAutomationDelegate	685
14.36.4.4 mValueString	685
14.37 AAX_CStateTaperDelegate< T > Class Template Reference	686
14.37.1 Description	686
14.37.2 Constructor & Destructor Documentation	687
14.37.2.1 AAX_CStateTaperDelegate()	687
14.37.3 Member Function Documentation	688
14.37.3.1 Clone()	688
14.37.3.2 GetMinimumValue()	688
14.37.3.3 GetMaximumValue()	688
14.37.3.4 ConstrainRealValue()	688
14.37.3.5 NormalizedToReal()	689
14.37.3.6 RealToNormalized()	689
14.38 AAX_CString Class Reference	690
14.38.1 Description	690
14.38.2 Constructor & Destructor Documentation	692
14.38.2.1 AAX_CString() [1/5]	692
14.38.2.2 AAX_CString() [2/5]	692
14.38.2.3 AAX_CString() [3/5]	692
14.38.2.4 AAX_CString() [4/5]	693
14.38.2.5 AAX_CString() [5/5]	693
14.38.3 Member Function Documentation	693
14.38.3.1 Length()	693
14.38.3.2 MaxLength()	694
14.38.3.3 Get()	694
14.38.3.4 Set()	695
14.38.3.5 operator=() [1/5]	695
14.38.3.6 operator=() [2/5]	695
14.38.3.7 AAX_DEFAULT_MOVE_CTOR()	695
14.38.3.8 StdString() [1/2]	696
14.38.3.9 StdString() [2/2]	696
14.38.3.10 operator=() [3/5]	696
14.38.3.11 operator=() [4/5]	696
14.38.3.12 operator=() [5/5]	696
14.38.3.13 Clear()	696
14.38.3.14 Empty()	697
14.38.3.15 Erase()	697
14.38.3.16 Append() [1/2]	697
14.38.3.17 Append() [2/2]	698

14.38.3.18 AppendNumber() [1/2]	698
14.38.3.19 AppendNumber() [2/2]	698
14.38.3.20 AppendHex()	698
14.38.3.21 Insert() [1/2]	699
14.38.3.22 Insert() [2/2]	699
14.38.3.23 InsertNumber() [1/2]	699
14.38.3.24 InsertNumber() [2/2]	699
14.38.3.25 InsertHex()	699
14.38.3.26 Replace() [1/2]	699
14.38.3.27 Replace() [2/2]	700
14.38.3.28 FindFirst() [1/3]	700
14.38.3.29 FindFirst() [2/3]	700
14.38.3.30 FindFirst() [3/3]	700
14.38.3.31 FindLast() [1/3]	700
14.38.3.32 FindLast() [2/3]	700
14.38.3.33 FindLast() [3/3]	700
14.38.3.34 CString()	701
14.38.3.35 ToDouble()	701
14.38.3.36 ToInteger()	701
14.38.3.37 SubString()	702
14.38.3.38 Equals() [1/3]	702
14.38.3.39 Equals() [2/3]	703
14.38.3.40 Equals() [3/3]	703
14.38.3.41 operator==() [1/3]	703
14.38.3.42 operator==() [2/3]	704
14.38.3.43 operator==() [3/3]	704
14.38.3.44 operator!=() [1/3]	704
14.38.3.45 operator!=() [2/3]	704
14.38.3.46 operator!=() [3/3]	704
14.38.3.47 operator<()	705
14.38.3.48 operator>()	705
14.38.3.49 operator[]() [1/2]	705
14.38.3.50 operator[]() [2/2]	705
14.38.3.51 operator+=() [1/3]	705
14.38.3.52 operator+=() [2/3]	705
14.38.3.53 operator+=() [3/3]	705
14.38.4 Friends And Related Function Documentation	706
14.38.4.1 operator<<	706
14.38.4.2 operator>>	706
14.38.5 Member Data Documentation	706
14.38.5.1 kInvalidIndex	706
14.38.5.2 kMaxStringLength	706

14.38.5.3 mString	706
14.39 AAX_CStringAbbreviations Class Reference	707
14.39.1 Description	707
14.39.2 Constructor & Destructor Documentation	707
14.39.2.1 AAX_CStringAbbreviations()	707
14.39.3 Member Function Documentation	707
14.39.3.1 SetPrimary()	707
14.39.3.2 Primary()	708
14.39.3.3 Add()	708
14.39.3.4 Get()	709
14.39.3.5 Clear()	709
14.40 AAX_CStringDataBuffer Class Reference	710
14.40.1 Description	711
14.40.2 Constructor & Destructor Documentation	711
14.40.2.1 AAX_CStringDataBuffer() [1/5]	712
14.40.2.2 AAX_CStringDataBuffer() [2/5]	712
14.40.2.3 AAX_CStringDataBuffer() [3/5]	712
14.40.2.4 AAX_CStringDataBuffer() [4/5]	712
14.40.2.5 AAX_CStringDataBuffer() [5/5]	712
14.40.2.6 ~AAX_CStringDataBuffer()	712
14.40.3 Member Function Documentation	712
14.40.3.1 operator=() [1/2]	713
14.40.3.2 operator=() [2/2]	713
14.40.3.3 Type()	713
14.40.3.4 Size()	713
14.40.3.5 Data()	713
14.41 AAX_CStringDataBufferOfType< T > Class Template Reference	714
14.41.1 Description	715
14.41.2 Constructor & Destructor Documentation	716
14.41.2.1 AAX_CStringDataBufferOfType() [1/5]	716
14.41.2.2 AAX_CStringDataBufferOfType() [2/5]	716
14.41.2.3 AAX_CStringDataBufferOfType() [3/5]	716
14.41.2.4 AAX_CStringDataBufferOfType() [4/5]	716
14.41.2.5 AAX_CStringDataBufferOfType() [5/5]	716
14.41.2.6 ~AAX_CStringDataBufferOfType()	716
14.41.3 Member Function Documentation	717
14.41.3.1 operator=() [1/2]	717
14.41.3.2 operator=() [2/2]	717
14.41.3.3 Type()	717
14.41.3.4 Size()	717
14.41.3.5 Data()	718
14.42 AAX_CStringDisplayDelegate< T > Class Template Reference	718

14.42.1 Description	719
14.42.2 Constructor & Destructor Documentation	720
14.42.2.1 AAX_CStringDisplayDelegate()	720
14.42.3 Member Function Documentation	720
14.42.3.1 Clone()	721
14.42.3.2 ValueToString() [1/2]	721
14.42.3.3 ValueToString() [2/2]	721
14.42.3.4 StringToValue()	722
14.42.4 Member Data Documentation	722
14.42.4.1 mStringMap	722
14.42.4.2 mInverseStringMap	723
14.43 AAX_CTaskAgent Class Reference	723
14.43.1 Description	724
14.43.2 Constructor & Destructor Documentation	725
14.43.2.1 AAX_CTaskAgent()	725
14.43.2.2 ~AAX_CTaskAgent()	726
14.43.3 Member Function Documentation	726
14.43.3.1 Initialize()	726
14.43.3.2 Uninitialize()	726
14.43.3.3 AddTask() [1/2]	726
14.43.3.4 CancelAllTasks()	727
14.43.3.5 AddTask() [2/2]	727
14.43.3.6 ReceiveTask()	727
14.43.3.7 GetController()	727
14.43.3.8 GetEffectParameters()	727
14.44 AAX_CTempoBreakpoint Struct Reference	728
14.44.1 Member Data Documentation	728
14.44.1.1 mSampleLocation	728
14.44.1.2 mValue	728
14.45 AAX_CUnitDisplayDelegateDecorator< T > Class Template Reference	728
14.45.1 Description	729
14.45.2 Constructor & Destructor Documentation	731
14.45.2.1 AAX_CUnitDisplayDelegateDecorator()	731
14.45.3 Member Function Documentation	731
14.45.3.1 Clone()	731
14.45.3.2 ValueToString() [1/2]	732
14.45.3.3 ValueToString() [2/2]	732
14.45.3.4 StringToValue()	733
14.45.4 Member Data Documentation	734
14.45.4.1 mUnitString	734
14.46 AAX_CUnitPrefixDisplayDelegateDecorator< T > Class Template Reference	734
14.46.1 Description	735

14.46.2 Constructor & Destructor Documentation	737
14.46.2.1 AAX_CUnitPrefixDisplayDelegateDecorator()	737
14.46.3 Member Function Documentation	737
14.46.3.1 Clone()	738
14.46.3.2 ValueToString() [1/2]	738
14.46.3.3 ValueToString() [2/2]	739
14.46.3.4 StringToValue()	739
14.47 AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT > Class Template Reference	740
14.47.1 Member Function Documentation	741
14.47.1.1 SetParameters()	741
14.47.1.2 DoTableLookupExtraFast() [1/2]	741
14.47.1.3 DoTableLookupExtraFastMulti()	742
14.47.1.4 DoTableLookupExtraFast() [2/2]	742
14.47.1.5 GetMin()	742
14.47.1.6 GetMaxMinusMin()	743
14.48 AAX_IACFAutomationDelegate Class Reference	743
14.48.1 Description	743
14.48.2 Member Function Documentation	744
14.48.2.1 RegisterParameter()	744
14.48.2.2 UnregisterParameter()	744
14.48.2.3 PostSetValueRequest()	745
14.48.2.4 PostCurrentValue()	745
14.48.2.5 PostTouchRequest()	746
14.48.2.6 PostReleaseRequest()	746
14.48.2.7 GetTouchState()	746
14.49 AAX_IACFCollection Class Reference	746
14.49.1 Description	747
14.49.2 Member Function Documentation	748
14.49.2.1 AddEffect()	748
14.49.2.2 SetManufacturerName()	748
14.49.2.3 AddPackageName()	748
14.49.2.4 SetPackageVersion()	749
14.49.2.5 SetProperties()	749
14.50 AAX_IACFComponentDescriptor Class Reference	749
14.50.1 Description	750
14.50.2 Member Function Documentation	751
14.50.2.1 Clear()	752
14.50.2.2 AddReservedField()	752
14.50.2.3 AddAudioIn()	752
14.50.2.4 AddAudioOut()	753
14.50.2.5 AddAudioBufferLength()	753
14.50.2.6 AddSampleRate()	753

14.50.2.7 AddClock()	754
14.50.2.8 AddSideChainIn()	754
14.50.2.9 AddDataInPort()	755
14.50.2.10 AddAuxOutputStem()	755
14.50.2.11 AddPrivateData()	756
14.50.2.12 AddDmaInstance()	756
14.50.2.13 AddMIDINode()	757
14.50.2.14 AddProcessProc_Native()	758
14.50.2.15 AddProcessProc_TI()	758
14.50.2.16 AddMeters()	759
14.51 AAX_IACFComponentDescriptor_V2 Class Reference	759
14.51.1 Description	760
14.51.2 Member Function Documentation	762
14.51.2.1 AddTemporaryData()	762
14.52 AAX_IACFComponentDescriptor_V3 Class Reference	762
14.52.1 Description	763
14.52.2 Member Function Documentation	765
14.52.2.1 AddProcessProc()	765
14.53 AAX_IACFController Class Reference	766
14.53.1 Description	767
14.53.2 Member Function Documentation	768
14.53.2.1 GetEffectID()	768
14.53.2.2 GetSampleRate()	768
14.53.2.3 GetInputStemFormat()	768
14.53.2.4 GetOutputStemFormat()	769
14.53.2.5 GetSignalLatency()	769
14.53.2.6 GetCycleCount()	769
14.53.2.7 GetTODLocation()	770
14.53.2.8 SetSignalLatency()	771
14.53.2.9 SetCycleCount()	771
14.53.2.10 PostPacket()	772
14.53.2.11 GetCurrentMeterValue()	773
14.53.2.12 GetMeterPeakValue()	773
14.53.2.13 ClearMeterPeakValue()	773
14.53.2.14 GetMeterClipped()	773
14.53.2.15 ClearMeterClipped()	774
14.53.2.16 GetMeterCount()	774
14.53.2.17 GetNextMIDIIPacket()	774
14.54 AAX_IACFController_V2 Class Reference	775
14.54.1 Description	776
14.54.2 Member Function Documentation	777
14.54.2.1 SendNotification()	778

14.54.2.2 GetHybridSignalLatency()	778
14.54.2.3 GetCurrentAutomationTimestamp()	779
14.54.2.4 GetHostName()	779
14.55 AAX_IACFController_V3 Class Reference	779
14.55.1 Description	780
14.55.2 Member Function Documentation	782
14.55.2.1 GetPlugInTargetPlatform()	782
14.55.2.2 GetIsAudioSuite()	782
14.56 AAX_IACFDataBuffer Class Reference	783
14.56.1 Description	783
14.56.2 Member Function Documentation	784
14.56.2.1 Type()	784
14.56.2.2 Size()	784
14.56.2.3 Data()	785
14.57 AAX_IACFDescriptionHost Class Reference	785
14.57.1 Description	786
14.57.2 Member Function Documentation	786
14.57.2.1 AcquireFeatureProperties()	786
14.58 AAX_IACFEffectDescriptor Class Reference	787
14.58.1 Description	787
14.58.2 Member Function Documentation	788
14.58.2.1 AddComponent()	788
14.58.2.2 AddName()	788
14.58.2.3 AddCategory()	789
14.58.2.4 AddCategoryBypassParameter()	789
14.58.2.5 AddProcPtr()	789
14.58.2.6 SetProperties()	791
14.58.2.7 AddResourceInfo()	791
14.58.2.8 AddMeterDescription()	791
14.59 AAX_IACFEffectDescriptor_V2 Class Reference	792
14.59.1 Description	792
14.59.2 Member Function Documentation	793
14.59.2.1 AddControlMIDINode()	793
14.60 AAX_IACFEffectDirectData Class Reference	794
14.60.1 Description	795
14.60.2 Member Function Documentation	796
14.60.2.1 Initialize()	796
14.60.2.2 Uninitialize()	796
14.60.2.3 TimerWakeup()	796
14.61 AAX_IACFEffectDirectData_V2 Class Reference	797
14.61.1 Member Function Documentation	798
14.61.1.1 NotificationReceived()	799

14.62 AAX_IACEffectGUI Class Reference	799
14.62.1 Description	800
14.62.2 Member Function Documentation	802
14.62.2.1 Initialize()	802
14.62.2.2 Uninitialize()	802
14.62.2.3 NotificationReceived()	803
14.62.2.4 SetViewContainer()	803
14.62.2.5 GetViewSize()	804
14.62.2.6 Draw()	804
14.62.2.7 TimerWakeup()	804
14.62.2.8 ParameterUpdated()	805
14.62.2.9 GetCustomLabel()	805
14.62.2.10 SetControlHighlightInfo()	805
14.63 AAX_IACEffectParameters Class Reference	806
14.63.1 Description	808
14.63.2 Member Function Documentation	812
14.63.2.1 Initialize()	812
14.63.2.2 Uninitialize()	812
14.63.2.3 NotificationReceived()	813
14.63.2.4 GetNumberOfParameters()	813
14.63.2.5 GetMasterBypassParameter()	814
14.63.2.6 GetParameterIsAutomatable()	814
14.63.2.7 GetParameterNumberOfSteps()	814
14.63.2.8 GetParameterName()	815
14.63.2.9 GetParameterNameOfLength()	815
14.63.2.10 GetParameterDefaultNormalizedValue()	816
14.63.2.11 SetParameterDefaultNormalizedValue()	816
14.63.2.12 GetParameterType()	816
14.63.2.13 GetParameterOrientation()	817
14.63.2.14 GetParameter()	817
14.63.2.15 GetParameterIndex()	818
14.63.2.16 GetParameterIDFromIndex()	818
14.63.2.17 GetParameterValueInfo()	819
14.63.2.18 GetParameterValueFromString()	819
14.63.2.19 GetParameterStringFromValue()	820
14.63.2.20 GetParameterValueString()	820
14.63.2.21 GetParameterNormalizedValue()	821
14.63.2.22 SetParameterNormalizedValue()	821
14.63.2.23 SetParameterNormalizedRelative()	822
14.63.2.24 TouchParameter()	822
14.63.2.25 ReleaseParameter()	823
14.63.2.26 UpdateParameterTouch()	823

14.63.2.27 UpdateParameterNormalizedValue()	824
14.63.2.28 UpdateParameterNormalizedRelative()	824
14.63.2.29 GenerateCoefficients()	825
14.63.2.30 ResetFieldData()	825
14.63.2.31 GetNumberOfChunks()	826
14.63.2.32 GetChunkIDFromIndex()	826
14.63.2.33 GetChunkSize()	826
14.63.2.34 GetChunk()	827
14.63.2.35 SetChunk()	828
14.63.2.36 CompareActiveChunk()	828
14.63.2.37 GetNumberOfChanges()	829
14.63.2.38 TimerWakeup()	829
14.63.2.39 GetCustomData()	829
14.63.2.40 SetCustomData()	830
14.63.2.41 DoMIDITransfers()	830
14.64 AAX_IACFEffEffectParameters_V2 Class Reference	830
14.64.1 Description	832
14.64.2 Member Function Documentation	835
14.64.2.1 UpdateMIDINodes()	835
14.64.2.2 UpdateControlMIDINodes()	836
14.65 AAX_IACFEffEffectParameters_V3 Class Reference	836
14.65.1 Description	838
14.66 AAX_IACFEffEffectParameters_V4 Class Reference	842
14.66.1 Description	843
14.66.2 Member Function Documentation	847
14.66.2.1 UpdatePageTable()	847
14.67 AAX_IACFFeatureInfo Class Reference	848
14.67.1 Description	848
14.67.2 Member Function Documentation	849
14.67.2.1 SupportLevel()	849
14.67.2.2 AcquireProperties()	850
14.68 AAX_IACFHostProcessor Class Reference	850
14.68.1 Description	851
14.68.2 Member Function Documentation	852
14.68.2.1 Initialize()	852
14.68.2.2 Uninitialize()	852
14.68.2.3 InitOutputBounds()	852
14.68.2.4 SetLocation()	853
14.68.2.5 RenderAudio()	854
14.68.2.6 PreRender()	854
14.68.2.7 PostRender()	855
14.68.2.8 AnalyzeAudio()	855

14.68.2.9 PreAnalyze()	856
14.68.2.10 PostAnalyze()	856
14.69 AAX_IACFHostProcessor_V2 Class Reference	857
14.69.1 Description	858
14.69.2 Member Function Documentation	859
14.69.2.1 GetClipNameSuffix()	859
14.70 AAX_IACFHostProcessorDelegate Class Reference	859
14.70.1 Description	860
14.70.2 Member Function Documentation	860
14.70.2.1 GetAudio()	861
14.70.2.2 GetSideChainInputNum()	861
14.71 AAX_IACFHostProcessorDelegate_V2 Class Reference	862
14.71.1 Description	862
14.71.2 Member Function Documentation	863
14.71.2.1 ForceAnalyze()	863
14.72 AAX_IACFHostProcessorDelegate_V3 Class Reference	864
14.72.1 Description	865
14.72.2 Member Function Documentation	865
14.72.2.1 ForceProcess()	865
14.73 AAX_IACFHostServices Class Reference	866
14.73.1 Description	866
14.73.2 Member Function Documentation	867
14.73.2.1 Assert()	867
14.73.2.2 Trace()	867
14.74 AAX_IACFHostServices_V2 Class Reference	868
14.74.1 Description	869
14.74.2 Member Function Documentation	869
14.74.2.1 StackTrace()	869
14.75 AAX_IACFHostServices_V3 Class Reference	870
14.75.1 Description	871
14.75.2 Member Function Documentation	872
14.75.2.1 HandleAssertFailure()	872
14.76 AAX_IACFPageTable Class Reference	872
14.76.1 Description	873
14.76.2 Member Function Documentation	874
14.76.2.1 Clear()	874
14.76.2.2 Empty()	874
14.76.2.3 GetNumPages()	875
14.76.2.4 InsertPage()	875
14.76.2.5 RemovePage()	875
14.76.2.6 GetNumMappedParameterIDs()	876
14.76.2.7 ClearMappedParameter()	876

14.76.2.8	GetMappedParameterID()	877
14.76.2.9	MapParameterID()	877
14.77	AAX_IACFPageTable_V2 Class Reference	878
14.77.1	Description	878
14.77.2	Member Function Documentation	879
14.77.2.1	GetNumParametersWithNameVariations()	880
14.77.2.2	GetNameVariationParameterIDAtIndex()	880
14.77.2.3	GetNumNameVariationsForParameter()	881
14.77.2.4	GetParameterNameVariationAtIndex()	881
14.77.2.5	GetParameterNameVariationOfLength()	882
14.77.2.6	ClearParameterNameVariations()	883
14.77.2.7	ClearNameVariationsForParameter()	883
14.77.2.8	SetParameterNameVariation()	884
14.78	AAX_IACFPageTableController Class Reference	884
14.78.1	Description	885
14.78.2	Member Function Documentation	886
14.78.2.1	CopyTableForEffect()	886
14.78.2.2	CopyTableOfLayoutForEffect()	887
14.79	AAX_IACFPageTableController_V2 Class Reference	888
14.79.1	Description	888
14.79.2	Member Function Documentation	889
14.79.2.1	CopyTableForEffectFromFile()	889
14.79.2.2	CopyTableOfLayoutFromFile()	890
14.80	AAX_IACFPrivateDataAccess Class Reference	891
14.80.1	Description	891
14.80.2	Member Function Documentation	892
14.80.2.1	ReadPortDirect()	892
14.80.2.2	WritePortDirect()	893
14.81	AAX_IACFPropertyMap Class Reference	893
14.81.1	Description	894
14.81.2	Member Function Documentation	894
14.81.2.1	GetProperty()	894
14.81.2.2	AddProperty()	895
14.81.2.3	RemoveProperty()	895
14.82	AAX_IACFPropertyMap_V2 Class Reference	895
14.82.1	Description	896
14.82.2	Member Function Documentation	897
14.82.2.1	AddPropertyWithIDArray()	897
14.82.2.2	GetPropertyWithIDArray()	897
14.83	AAX_IACFPropertyMap_V3 Class Reference	898
14.83.1	Description	899
14.83.2	Member Function Documentation	900

14.83.2.1	GetProperty64()	900
14.83.2.2	AddProperty64()	900
14.84	AAX_IACFSessionDocument Class Reference	901
14.84.1	Description	901
14.84.2	Member Function Documentation	902
14.84.2.1	GetDocumentData()	902
14.85	AAX_IACFSessionDocumentClient Class Reference	902
14.85.1	Description	903
14.85.2	Member Function Documentation	904
14.85.2.1	Initialize()	904
14.85.2.2	Uninitialize()	904
14.85.2.3	SetSessionDocument()	904
14.85.2.4	NotificationReceived()	905
14.86	AAX_IACFTask Class Reference	905
14.86.1	Description	906
14.86.2	Member Function Documentation	907
14.86.2.1	GetType()	907
14.86.2.2	GetArgumentOfType()	907
14.86.2.3	SetProgress()	908
14.86.2.4	GetProgress()	908
14.86.2.5	AddResult()	908
14.86.2.6	SetDone()	908
14.87	AAX_IACFTaskAgent Class Reference	909
14.87.1	Description	910
14.87.2	Member Function Documentation	910
14.87.2.1	Initialize()	910
14.87.2.2	Uninitialize()	911
14.87.2.3	AddTask()	911
14.87.2.4	CancelAllTasks()	911
14.88	AAX_IACFTransport Class Reference	912
14.88.1	Description	912
14.88.2	Member Function Documentation	913
14.88.2.1	GetCurrentTempo()	913
14.88.2.2	GetCurrentMeter()	914
14.88.2.3	IsTransportPlaying()	914
14.88.2.4	GetCurrentTickPosition()	914
14.88.2.5	GetCurrentLoopPosition()	915
14.88.2.6	GetCurrentNativeSampleLocation()	915
14.88.2.7	GetCustomTickPosition()	916
14.88.2.8	GetBarBeatPosition()	916
14.88.2.9	GetTicksPerQuarter()	917
14.88.2.10	GetCurrentTicksPerBeat()	917

14.89 AAX_IACFTransport_V2 Class Reference	917
14.89.1 Description	918
14.89.2 Member Function Documentation	919
14.89.2.1 GetTimelineSelectionStartPosition()	920
14.89.2.2 GetTimeCodeInfo()	920
14.89.2.3 GetFeetFramesInfo()	920
14.89.2.4 IsMetronomeEnabled()	921
14.90 AAX_IACFTransport_V3 Class Reference	921
14.90.1 Description	923
14.90.2 Member Function Documentation	924
14.90.2.1 GetHDTTimeCodeInfo()	924
14.91 AAX_IACFTransport_V4 Class Reference	924
14.91.1 Description	926
14.91.2 Member Function Documentation	927
14.91.2.1 GetTimelineSelectionEndPosition()	927
14.92 AAX_IACFTransportControl Class Reference	928
14.92.1 Description	928
14.92.2 Member Function Documentation	929
14.92.2.1 RequestTransportStart()	929
14.92.2.2 RequestTransportStop()	929
14.93 AAX_IACFViewContainer Class Reference	930
14.93.1 Description	930
14.93.2 Member Function Documentation	931
14.93.2.1 GetType()	931
14.93.2.2 GetPtr()	932
14.93.2.3 GetModifiers()	932
14.93.2.4 SetViewSize()	932
14.93.2.5 HandleParameterMouseDown()	933
14.93.2.6 HandleParameterMouseDownDrag()	933
14.93.2.7 HandleParameterMouseUp()	933
14.94 AAX_IACFViewContainer_V2 Class Reference	934
14.94.1 Description	935
14.94.2 Member Function Documentation	936
14.94.2.1 HandleMultipleParametersMouseDown()	936
14.94.2.2 HandleMultipleParametersMouseDownDrag()	936
14.94.2.3 HandleMultipleParametersMouseUp()	937
14.95 AAX_IACFViewContainer_V3 Class Reference	937
14.95.1 Description	938
14.95.2 Member Function Documentation	940
14.95.2.1 HandleParameterMouseEnter()	940
14.95.2.2 HandleParameterMouseExit()	940
14.96 AAX_IAutomationDelegate Class Reference	940

14.96.1 Description	941
14.96.2 Constructor & Destructor Documentation	941
14.96.2.1 ~AAX_IAutomationDelegate()	942
14.96.3 Member Function Documentation	942
14.96.3.1 RegisterParameter()	942
14.96.3.2 UnregisterParameter()	943
14.96.3.3 PostSetValueRequest()	943
14.96.3.4 PostCurrentValue()	944
14.96.3.5 PostTouchRequest()	944
14.96.3.6 PostReleaseRequest()	945
14.96.3.7 GetTouchState()	945
14.96.3.8 ParameterNameChanged()	945
14.97 AAX_ICollection Class Reference	946
14.97.1 Description	947
14.97.2 Constructor & Destructor Documentation	947
14.97.2.1 ~AAX_ICollection()	947
14.97.3 Member Function Documentation	947
14.97.3.1 NewDescriptor()	948
14.97.3.2 AddEffect()	948
14.97.3.3 SetManufacturerName()	948
14.97.3.4 AddPackageName()	949
14.97.3.5 SetPackageVersion()	949
14.97.3.6 NewPropertyMap()	949
14.97.3.7 SetProperties()	949
14.97.3.8 DescriptionHost() [1/2]	950
14.97.3.9 DescriptionHost() [2/2]	950
14.97.3.10 HostDefinition()	951
14.98 AAX_IComponentDescriptor Class Reference	951
14.98.1 Description	951
14.98.2 Constructor & Destructor Documentation	953
14.98.2.1 ~AAX_IComponentDescriptor()	953
14.98.3 Member Function Documentation	953
14.98.3.1 Clear()	953
14.98.3.2 AddAudioIn()	953
14.98.3.3 AddAudioOut()	954
14.98.3.4 AddAudioBufferLength()	955
14.98.3.5 AddSampleRate()	956
14.98.3.6 AddClock()	956
14.98.3.7 AddSideChainIn()	957
14.98.3.8 AddDataInPort()	957
14.98.3.9 AddAuxOutputStem()	958
14.98.3.10 AddPrivateData()	959

14.98.3.11 AddTemporaryData()	960
14.98.3.12 AddDmaInstance()	960
14.98.3.13 AddMeters()	961
14.98.3.14 AddMIDINode()	962
14.98.3.15 AddReservedField()	963
14.98.3.16 NewPropertyMap()	963
14.98.3.17 DuplicatePropertyMap()	964
14.98.3.18 AddProcessProc_Native() [1/2]	964
14.98.3.19 AddProcessProc_TI()	965
14.98.3.20 AddProcessProc()	966
14.98.3.21 AddProcessProc_Native() [2/2]	967
14.99 AAX_IContainer Class Reference	968
14.99.1 Description	968
14.99.2 Member Enumeration Documentation	969
14.99.2.1 EStatus	969
14.99.3 Constructor & Destructor Documentation	969
14.99.3.1 ~AAX_IContainer()	969
14.99.4 Member Function Documentation	969
14.99.4.1 Clear()	969
14.100 AAX_IController Class Reference	970
14.100.1 Description	970
14.100.2 Constructor & Destructor Documentation	972
14.100.2.1 ~AAX_IController()	972
14.100.3 Member Function Documentation	972
14.100.3.1 GetEffectID()	972
14.100.3.2 GetSampleRate()	972
14.100.3.3 GetInputStemFormat()	973
14.100.3.4 GetOutputStemFormat()	973
14.100.3.5 GetSignalLatency()	973
14.100.3.6 GetCycleCount()	974
14.100.3.7 GetTODLocation()	975
14.100.3.8 SetSignalLatency()	975
14.100.3.9 SetCycleCount()	976
14.100.3.10 PostPacket()	976
14.100.3.11 SendNotification() [1/2]	977
14.100.3.12 SendNotification() [2/2]	978
14.100.3.13 GetCurrentMeterValue()	978
14.100.3.14 GetMeterPeakValue()	978
14.100.3.15 ClearMeterPeakValue()	979
14.100.3.16 GetMeterCount()	979
14.100.3.17 GetMeterClipped()	979
14.100.3.18 ClearMeterClipped()	981

14.100.3.19	GetNextMIDIPacket()	981
14.100.3.20	GetCurrentAutomationTimestamp()	981
14.100.3.21	GetHostName()	982
14.100.3.22	GetPlugInTargetPlatform()	982
14.100.3.23	GetIsAudioSuite()	983
14.100.3.24	CreateTableCopyForEffect()	983
14.100.3.25	CreateTableCopyForLayout()	984
14.100.3.26	CreateTableCopyForEffectFromFile()	984
14.100.3.27	CreateTableCopyForLayoutFromFile()	985
14.101	AAX_IDataBuffer Class Reference	986
14.101.1	Description	986
14.101.2	Member Function Documentation	987
14.101.2.1	ACF_DECLARE_STANDARD_UNKNOWN()	987
14.101.2.2	AAX_DELETE()	987
14.101.3	Member Data Documentation	987
14.101.3.1	AAX_OVERRIDE	988
14.102	AAX_IDataBufferWrapper Class Reference	988
14.102.1	Description	988
14.102.2	Constructor & Destructor Documentation	989
14.102.2.1	~AAX_IDataBufferWrapper()	989
14.102.3	Member Function Documentation	989
14.102.3.1	Type()	989
14.102.3.2	Size()	989
14.102.3.3	Data()	989
14.103	AAX_IDescriptionHost Class Reference	990
14.103.1	Description	990
14.103.2	Constructor & Destructor Documentation	990
14.103.2.1	~AAX_IDescriptionHost()	990
14.103.3	Member Function Documentation	990
14.103.3.1	AcquireFeatureProperties()	991
14.104	AAX_IDisplayDelegate< T > Class Template Reference	991
14.104.1	Description	992
14.104.2	Display delegate decorators	992
14.104.2.1	Display delegate decorator implementation	993
14.104.2.2	Decibel decorator example	993
14.104.3	Member Function Documentation	994
14.104.3.1	Clone()	994
14.104.3.2	ValueToString() [1/2]	994
14.104.3.3	ValueToString() [2/2]	995
14.104.3.4	StringToValue()	995
14.105	AAX_IDisplayDelegateBase Class Reference	996
14.105.1	Description	996

14.105.2 Constructor & Destructor Documentation	997
14.105.2.1 ~AAX_IDisplayDelegateBase()	997
14.106 AAX_IDisplayDelegateDecorator< T > Class Template Reference	997
14.106.1 Description	998
14.106.2 Constructor & Destructor Documentation	999
14.106.2.1 AAX_IDisplayDelegateDecorator() [1/2]	999
14.106.2.2 AAX_IDisplayDelegateDecorator() [2/2]	999
14.106.2.3 ~AAX_IDisplayDelegateDecorator()	1000
14.106.3 Member Function Documentation	1000
14.106.3.1 Clone()	1000
14.106.3.2 ValueToString() [1/2]	1001
14.106.3.3 ValueToString() [2/2]	1002
14.106.3.4 StringToValue()	1002
14.107 AAX_IDma Class Reference	1003
14.107.1 Description	1003
14.107.2 Member Enumeration Documentation	1005
14.107.2.1 EState	1005
14.107.2.2 EMode	1005
14.107.3 Constructor & Destructor Documentation	1007
14.107.3.1 ~AAX_IDma()	1007
14.107.4 Member Function Documentation	1007
14.107.4.1 PostRequest()	1007
14.107.4.2 IsTransferComplete()	1008
14.107.4.3 SetDmaState()	1008
14.107.4.4 GetDmaState()	1008
14.107.4.5 GetDmaMode()	1008
14.107.4.6 SetSrc()	1008
14.107.4.7 GetSrc()	1009
14.107.4.8 SetDst()	1009
14.107.4.9 GetDst()	1009
14.107.4.10 SetBurstLength()	1010
14.107.4.11 GetBurstLength()	1010
14.107.4.12 SetNumBursts()	1010
14.107.4.13 GetNumBursts()	1011
14.107.4.14 SetTransferSize()	1011
14.107.4.15 GetTransferSize()	1011
14.107.4.16 SetFifoBuffer()	1011
14.107.4.17 GetFifoBuffer()	1012
14.107.4.18 SetLinearBuffer()	1012
14.107.4.19 GetLinearBuffer()	1012
14.107.4.20 SetOffsetTable()	1012
14.107.4.21 GetOffsetTable()	1013

14.107.4.22 SetNumOffsets()	1013
14.107.4.23 GetNumOffsets()	1013
14.107.4.24 SetBaseOffset()	1013
14.107.4.25 GetBaseOffset()	1014
14.107.4.26 SetFifoSize()	1014
14.107.4.27 GetFifoSize()	1014
14.108 AAX_IEffectDescriptor Class Reference	1014
14.108.1 Description	1015
14.108.2 Constructor & Destructor Documentation	1015
14.108.2.1 ~AAX_IEffectDescriptor()	1015
14.108.3 Member Function Documentation	1016
14.108.3.1 NewComponentDescriptor()	1016
14.108.3.2 AddComponent()	1016
14.108.3.3 AddName()	1017
14.108.3.4 AddCategory()	1017
14.108.3.5 AddCategoryBypassParameter()	1018
14.108.3.6 AddProcPtr()	1018
14.108.3.7 NewPropertyMap()	1018
14.108.3.8 SetProperties()	1018
14.108.3.9 AddResourceInfo()	1019
14.108.3.10 AddMeterDescription()	1019
14.108.3.11 AddControlMIDIINode()	1019
14.109 AAX_IEffectDirectData Class Reference	1020
14.109.1 Description	1022
14.109.2 Member Function Documentation	1023
14.109.2.1 ACF_DECLARE_STANDARD_UNKNOWN()	1023
14.109.2.2 AAX_DELETE()	1023
14.109.3 Member Data Documentation	1023
14.109.3.1 override	1023
14.110 AAX_IEffectGUI Class Reference	1024
14.110.1 Description	1024
14.110.2 Member Function Documentation	1026
14.110.2.1 ACF_DECLARE_STANDARD_UNKNOWN()	1026
14.110.2.2 AAX_DELETE()	1026
14.110.3 Member Data Documentation	1026
14.110.3.1 override	1026
14.111 AAX_IEffectParameters Class Reference	1027
14.111.1 Description	1028
14.111.2 Related classes	1029
14.111.3 Member Function Documentation	1033
14.111.3.1 ACF_DECLARE_STANDARD_UNKNOWN()	1033
14.111.3.2 AAX_DELETE()	1033

14.111.4 Member Data Documentation	1033
14.111.4.1 override	1033
14.112 AAX_IFeatureInfo Class Reference	1033
14.112.1 Constructor & Destructor Documentation	1034
14.112.1.1 ~AAX_IFeatureInfo()	1034
14.112.2 Member Function Documentation	1034
14.112.2.1 SupportLevel()	1034
14.112.2.2 AcquireProperties()	1034
14.112.2.3 ID()	1035
14.113 AAX_IHostProcessor Class Reference	1035
14.113.1 Description	1036
14.113.2 Member Function Documentation	1037
14.113.2.1 ACF_DECLARE_STANDARD_UNKNOWN()	1037
14.113.2.2 AAX_DELETE()	1038
14.113.3 Member Data Documentation	1038
14.113.3.1 override	1038
14.114 AAX_IHostProcessorDelegate Class Reference	1038
14.114.1 Description	1038
14.114.2 Constructor & Destructor Documentation	1039
14.114.2.1 ~AAX_IHostProcessorDelegate()	1039
14.114.3 Member Function Documentation	1039
14.114.3.1 GetAudio()	1039
14.114.3.2 GetSideChainInputNum()	1040
14.114.3.3 ForceAnalyze()	1040
14.114.3.4 ForceProcess()	1041
14.115 AAX_IHostServices Class Reference	1041
14.115.1 Description	1041
14.115.2 Constructor & Destructor Documentation	1042
14.115.2.1 ~AAX_IHostServices()	1042
14.115.3 Member Function Documentation	1042
14.115.3.1 HandleAssertFailure()	1042
14.115.3.2 Trace()	1043
14.115.3.3 StackTrace()	1043
14.116 AAX_IMIDIMessageInfoDelegate Class Reference	1044
14.116.1 Constructor & Destructor Documentation	1044
14.116.1.1 ~AAX_IMIDIMessageInfoDelegate()	1044
14.116.2 Member Function Documentation	1044
14.116.2.1 Mask()	1044
14.116.2.2 Length()	1045
14.116.2.3 ToString()	1045
14.116.2.4 Accepts()	1045
14.116.2.5 Accepts_ExactStatus()	1046

14.116.2.6 ToString_AppendNumber()	1046
14.116.2.7 ToString_AppendCStr()	1047
14.116.2.8 ToString_AppendByteRange()	1047
14.116.2.9 ToString_AppendValid()	1048
14.117 AAX_IMIDINode Class Reference	1048
14.117.1 Description	1048
14.117.2 Constructor & Destructor Documentation	1049
14.117.2.1 ~AAX_IMIDINode()	1049
14.117.3 Member Function Documentation	1049
14.117.3.1 GetNodeBuffer()	1049
14.117.3.2 PostMIDIpacket()	1049
14.117.3.3 GetTransport()	1050
14.118 AAX_IPacketHandler Struct Reference	1050
14.118.1 Description	1050
14.118.2 Constructor & Destructor Documentation	1051
14.118.2.1 ~AAX_IPacketHandler()	1051
14.118.3 Member Function Documentation	1051
14.118.3.1 Clone()	1051
14.118.3.2 Call()	1051
14.119 AAX_IPageTable Class Reference	1051
14.119.1 Description	1052
14.119.2 Constructor & Destructor Documentation	1052
14.119.2.1 ~AAX_IPageTable()	1053
14.119.3 Member Function Documentation	1053
14.119.3.1 Clear()	1053
14.119.3.2 Empty()	1053
14.119.3.3 GetNumPages()	1053
14.119.3.4 InsertPage()	1055
14.119.3.5 RemovePage()	1055
14.119.3.6 GetNumMappedParameterIDs()	1056
14.119.3.7 ClearMappedParameter()	1056
14.119.3.8 GetMappedParameterID()	1057
14.119.3.9 MapParameterID()	1057
14.119.3.10 GetNumParametersWithNameVariations()	1058
14.119.3.11 GetNameVariationParameterIDAtIndex()	1058
14.119.3.12 GetNumNameVariationsForParameter()	1059
14.119.3.13 GetParameterNameVariationAtIndex()	1059
14.119.3.14 GetParameterNameVariationOfLength()	1060
14.119.3.15 ClearParameterNameVariations()	1061
14.119.3.16 ClearNameVariationsForParameter()	1061
14.119.3.17 SetParameterNameVariation()	1062
14.120 AAX_IParameter Class Reference	1063

14.120.1 Description	1063
14.120.2 Constructor & Destructor Documentation	1066
14.120.2.1 ~AAX_IParameter()	1066
14.120.3 Member Function Documentation	1066
14.120.3.1 CloneValue()	1066
14.120.3.2 Identifier()	1067
14.120.3.3 SetName()	1067
14.120.3.4 Name()	1067
14.120.3.5 AddShortenedName()	1068
14.120.3.6 ShortenedName()	1068
14.120.3.7 ClearShortenedNames()	1068
14.120.3.8 Automatable()	1069
14.120.3.9 SetAutomationDelegate()	1069
14.120.3.10 Touch()	1069
14.120.3.11 Release()	1070
14.120.3.12 SetNormalizedValue()	1070
14.120.3.13 GetNormalizedValue()	1070
14.120.3.14 SetNormalizedDefaultValue()	1070
14.120.3.15 GetNormalizedDefaultValue()	1071
14.120.3.16 SetToDefaultValue()	1071
14.120.3.17 SetNumberOfSteps()	1071
14.120.3.18 GetNumberOfSteps()	1071
14.120.3.19 GetStepValue()	1072
14.120.3.20 GetNormalizedValueFromStep()	1072
14.120.3.21 GetStepValueFromNormalizedValue()	1072
14.120.3.22 SetStepValue()	1072
14.120.3.23 GetValueString() [1/2]	1072
14.120.3.24 GetValueString() [2/2]	1073
14.120.3.25 GetNormalizedValueFromBool()	1073
14.120.3.26 GetNormalizedValueFromInt32()	1074
14.120.3.27 GetNormalizedValueFromFloat()	1074
14.120.3.28 GetNormalizedValueFromDouble()	1075
14.120.3.29 GetNormalizedValueFromString()	1075
14.120.3.30 GetBoolFromNormalizedValue()	1076
14.120.3.31 GetInt32FromNormalizedValue()	1076
14.120.3.32 GetFloatFromNormalizedValue()	1076
14.120.3.33 GetDoubleFromNormalizedValue()	1077
14.120.3.34 GetStringFromNormalizedValue() [1/2]	1077
14.120.3.35 GetStringFromNormalizedValue() [2/2]	1078
14.120.3.36 SetValueFromString()	1078
14.120.3.37 GetValueAsBool()	1079
14.120.3.38 GetValueAsInt32()	1079

14.120.3.39	GetValueAsFloat()	1080
14.120.3.40	GetValueAsDouble()	1080
14.120.3.41	GetValueAsString()	1080
14.120.3.42	SetValueWithBool()	1081
14.120.3.43	SetValueWithInt32()	1081
14.120.3.44	SetValueWithFloat()	1082
14.120.3.45	SetValueWithDouble()	1082
14.120.3.46	SetValueWithString()	1082
14.120.3.47	SetType()	1083
14.120.3.48	GetType()	1083
14.120.3.49	SetOrientation()	1083
14.120.3.50	GetOrientation()	1084
14.120.3.51	SetTaperDelegate()	1084
14.120.3.52	SetDisplayDelegate()	1084
14.120.3.53	UpdateNormalizedValue()	1085
14.121	AAX_IParаметerValue Class Reference	1085
14.121.1	Description	1085
14.121.2	Constructor & Destructor Documentation	1086
14.121.2.1	~AAX_IParаметerValue()	1086
14.121.3	Member Function Documentation	1086
14.121.3.1	Clone()	1086
14.121.3.2	Identifier()	1087
14.121.3.3	GetValueAsBool()	1087
14.121.3.4	GetValueAsInt32()	1087
14.121.3.5	GetValueAsFloat()	1088
14.121.3.6	GetValueAsDouble()	1088
14.121.3.7	GetValueAsString()	1088
14.122	AAX_IPointerQueue< T > Class Template Reference	1089
14.122.1	Description	1090
14.122.2	Member Typedef Documentation	1090
14.122.2.1	template_type	1090
14.122.2.2	value_type	1091
14.122.3	Constructor & Destructor Documentation	1091
14.122.3.1	~AAX_IPointerQueue()	1091
14.122.4	Member Function Documentation	1091
14.122.4.1	Clear()	1091
14.122.4.2	Push()	1091
14.122.4.3	Pop()	1092
14.122.4.4	Peek()	1092
14.123	AAX_IPrivateDataAccess Class Reference	1092
14.123.1	Description	1093
14.123.2	Constructor & Destructor Documentation	1093

14.123.2.1 ~AAX_IPrivateDataAccess()	1093
14.123.3 Member Function Documentation	1093
14.123.3.1 ReadPortDirect()	1093
14.123.3.2 WritePortDirect()	1094
14.124 AAX_IPropertyMap Class Reference	1094
14.124.1 Description	1095
14.124.2 Constructor & Destructor Documentation	1096
14.124.2.1 ~AAX_IPropertyMap()	1096
14.124.3 Member Function Documentation	1096
14.124.3.1 GetProperty()	1096
14.124.3.2 GetPointerProperty()	1096
14.124.3.3 AddProperty()	1097
14.124.3.4 AddPointerProperty() [1/2]	1097
14.124.3.5 AddPointerProperty() [2/2]	1098
14.124.3.6 RemoveProperty()	1098
14.124.3.7 AddPropertyWithIDArray()	1099
14.124.3.8 GetPropertyWithIDArray()	1099
14.124.3.9 GetUnknown()	1099
14.125 AAX_ISessionDocument Class Reference	1100
14.125.1 Description	1100
14.125.2 Constructor & Destructor Documentation	1101
14.125.2.1 ~AAX_ISessionDocument()	1101
14.125.3 Member Function Documentation	1101
14.125.3.1 Valid()	1101
14.125.3.2 GetTempoMap()	1101
14.125.3.3 GetDocumentData()	1101
14.126 AAX_ISessionDocumentClient Class Reference	1102
14.126.1 Description	1103
14.126.2 Member Function Documentation	1104
14.126.2.1 ACF_DECLARE_STANDARD_UNKNOWN()	1104
14.126.2.2 AAX_DELETE()	1104
14.126.3 Member Data Documentation	1104
14.126.3.1 override	1104
14.127 AAX_IString Class Reference	1104
14.127.1 Description	1105
14.127.2 Constructor & Destructor Documentation	1105
14.127.2.1 ~AAX_IString()	1105
14.127.3 Member Function Documentation	1105
14.127.3.1 Length()	1105
14.127.3.2 MaxLength()	1106
14.127.3.3 Get()	1106
14.127.3.4 Set()	1106

14.127.3.5 operator=() [1/2]	1106
14.127.3.6 operator=() [2/2]	1107
14.128 AAX_ITaperDelegate< T > Class Template Reference	1107
14.128.1 Description	1108
14.128.2 Member Function Documentation	1108
14.128.2.1 Clone()	1109
14.128.2.2 GetMaximumValue()	1109
14.128.2.3 GetMinimumValue()	1109
14.128.2.4 ConstrainRealValue()	1110
14.128.2.5 NormalizedToReal()	1110
14.128.2.6 RealToNormalized()	1110
14.129 AAX_ITaperDelegateBase Class Reference	1111
14.129.1 Description	1111
14.129.2 Constructor & Destructor Documentation	1112
14.129.2.1 ~AAX_ITaperDelegateBase()	1112
14.130 AAX_ITask Class Reference	1113
14.130.1 Description	1113
14.130.2 Constructor & Destructor Documentation	1113
14.130.2.1 ~AAX_ITask()	1114
14.130.3 Member Function Documentation	1114
14.130.3.1 GetType()	1114
14.130.3.2 GetArgumentOfType()	1114
14.130.3.3 SetProgress()	1114
14.130.3.4 GetProgress()	1115
14.130.3.5 AddResult()	1115
14.130.3.6 SetDone()	1115
14.131 AAX_ITaskAgent Class Reference	1116
14.131.1 Description	1117
14.131.2 Member Function Documentation	1118
14.131.2.1 ACF_DECLARE_STANDARD_UNKNOWN()	1118
14.131.2.2 AAX_DELETE()	1118
14.131.3 Member Data Documentation	1118
14.131.3.1 AAX_OVERRIDE	1118
14.132 AAX_ITransport Class Reference	1119
14.132.1 Description	1119
14.132.2 Constructor & Destructor Documentation	1120
14.132.2.1 ~AAX_ITransport()	1120
14.132.3 Member Function Documentation	1120
14.132.3.1 GetCurrentTempo()	1121
14.132.3.2 GetCurrentMeter()	1121
14.132.3.3 IsTransportPlaying()	1121
14.132.3.4 GetCurrentTickPosition()	1122

14.132.3.5	GetCurrentLoopPosition()	1122
14.132.3.6	GetCurrentNativeSampleLocation()	1123
14.132.3.7	GetCustomTickPosition()	1123
14.132.3.8	GetBarBeatPosition()	1124
14.132.3.9	GetTicksPerQuarter()	1124
14.132.3.10	GetCurrentTicksPerBeat()	1124
14.132.3.11	GetTimelineSelectionStartPosition()	1125
14.132.3.12	GetTimeCodeInfo()	1125
14.132.3.13	GetFeetFramesInfo()	1126
14.132.3.14	IsMetronomeEnabled()	1126
14.132.3.15	GetHDTTimeCodeInfo()	1126
14.132.3.16	RequestTransportStart()	1127
14.132.3.17	RequestTransportStop()	1127
14.132.3.18	GetTimelineSelectionEndPosition()	1127
14.133	AAX_IViewContainer Class Reference	1128
14.133.1	Description	1128
14.133.2	Constructor & Destructor Documentation	1129
14.133.2.1	~AAX_IViewContainer()	1129
14.133.3	Member Function Documentation	1130
14.133.3.1	GetType()	1130
14.133.3.2	GetPtr()	1130
14.133.3.3	GetModifiers()	1130
14.133.3.4	SetViewSize()	1131
14.133.3.5	HandleParameterMouseDown()	1131
14.133.3.6	HandleParameterMouseDrag()	1131
14.133.3.7	HandleParameterMouseUp()	1132
14.133.3.8	HandleParameterMouseEnter()	1132
14.133.3.9	HandleParameterMouseExit()	1133
14.133.3.10	HandleMultipleParametersMouseDown()	1133
14.133.3.11	HandleMultipleParametersMouseDrag()	1133
14.133.3.12	HandleMultipleParametersMouseUp()	1134
14.134	AAX_Map Class Reference	1134
14.134.1	Constructor & Destructor Documentation	1135
14.134.1.1	AAX_Map()	1135
14.134.1.2	~AAX_Map()	1135
14.134.2	Member Function Documentation	1135
14.134.2.1	SetCoefficients()	1135
14.134.2.2	GetCoefficient()	1135
14.134.2.3	GetUpperBoundIndex()	1136
14.134.2.4	GetX()	1136
14.134.2.5	GetY()	1136
14.134.2.6	GetFirstX()	1136

14.134.2.7 GetFirstY()	1136
14.134.2.8 GetLastX()	1136
14.134.2.9 GetLastY()	1136
14.134.2.10 GetSize()	1137
14.135 AAX_Point Struct Reference	1137
14.135.1 Description	1137
14.135.2 Constructor & Destructor Documentation	1137
14.135.2.1 AAX_Point() [1/2]	1137
14.135.2.2 AAX_Point() [2/2]	1137
14.135.3 Member Data Documentation	1138
14.135.3.1 vert	1138
14.135.3.2 horz	1138
14.136 AAX_Rect Struct Reference	1138
14.136.1 Description	1138
14.136.2 Constructor & Destructor Documentation	1138
14.136.2.1 AAX_Rect() [1/2]	1139
14.136.2.2 AAX_Rect() [2/2]	1139
14.136.3 Member Data Documentation	1139
14.136.3.1 top	1139
14.136.3.2 left	1139
14.136.3.3 width	1139
14.136.3.4 height	1140
14.137 AAX_SHybridRenderInfo Struct Reference	1140
14.137.1 Description	1140
14.137.2 Member Data Documentation	1140
14.137.2.1 mAudioInputs	1140
14.137.2.2 mNumAudioInputs	1140
14.137.2.3 mAudioOutputs	1141
14.137.2.4 mNumAudioOutputs	1141
14.137.2.5 mNumSamples	1141
14.137.2.6 mClock	1141
14.138 AAX_SInstrumentPrivateData Struct Reference	1141
14.138.1 Description	1141
14.138.2 Member Data Documentation	1142
14.138.2.1 mMonolithicParametersPtr	1142
14.139 AAX_SInstrumentRenderInfo Struct Reference	1142
14.139.1 Description	1142
14.139.2 Member Data Documentation	1143
14.139.2.1 mAudioInputs	1143
14.139.2.2 mAudioOutputs	1143
14.139.2.3 mNumSamples	1144
14.139.2.4 mClock	1144

14.139.2.5 mInputNode	1144
14.139.2.6 mGlobalNode	1144
14.139.2.7 mTransportNode	1144
14.139.2.8 mAdditionalInputMIDINodes	1144
14.139.2.9 mPrivateData	1145
14.139.2.10 mMeters	1145
14.139.2.11 mCurrentStateNum	1145
14.140 AAX_SInstrumentSetupInfo Struct Reference	1145
14.140.1 Description	1145
14.140.2 Constructor & Destructor Documentation	1147
14.140.2.1 AAX_SInstrumentSetupInfo()	1147
14.140.3 Member Data Documentation	1147
14.140.3.1 mNeedsGlobalMIDI	1147
14.140.3.2 mGlobalMIDINodeName	1147
14.140.3.3 mGlobalMIDIEventMask	1148
14.140.3.4 mNeedsInputMIDI	1148
14.140.3.5 mInputMIDINodeName	1148
14.140.3.6 mInputMIDIChannelMask	1148
14.140.3.7 mNumAdditionalInputMIDINodes	1149
14.140.3.8 mNeedsTransport	1149
14.140.3.9 mTransportMIDINodeName	1149
14.140.3.10 mNumMeters	1149
14.140.3.11 mMeterIDs	1150
14.140.3.12 mNumAuxOutputStems	1150
14.140.3.13 mAuxOutputStemNames	1150
14.140.3.14 mAuxOutputStemFormats	1150
14.140.3.15 mHybridInputStemFormat	1151
14.140.3.16 mHybridOutputStemFormat	1151
14.140.3.17 mInputStemFormat	1151
14.140.3.18 mOutputStemFormat	1151
14.140.3.19 mUseHostGeneratedGUI	1152
14.140.3.20 mCanBypass	1152
14.140.3.21 mManufacturerID	1152
14.140.3.22 mProductID	1152
14.140.3.23 mPluginID	1153
14.140.3.24 mAudiosuiteID	1153
14.140.3.25 mMultiMonoSupport	1153
14.141 AAX_SPluginChunk Struct Reference	1153
14.141.1 Description	1153
14.141.2 Member Data Documentation	1154
14.141.2.1 fSize	1154
14.141.2.2 fVersion	1154

14.141.2.3 fManufacturerID	1154
14.141.2.4 fProductID	1155
14.141.2.5 fPluginID	1155
14.141.2.6 fChunkID	1155
14.141.2.7 fName	1155
14.141.2.8 fData	1155
14.142 AAX_SPluginChunkHeader Struct Reference	1156
14.142.1 Description	1156
14.142.2 Member Data Documentation	1157
14.142.2.1 fSize	1157
14.142.2.2 fVersion	1157
14.142.2.3 fManufacturerID	1157
14.142.2.4 fProductID	1157
14.142.2.5 fPluginID	1157
14.142.2.6 fChunkID	1158
14.142.2.7 fName	1158
14.143 AAX_SPluginIdentifierTriad Struct Reference	1158
14.143.1 Description	1158
14.143.2 Member Data Documentation	1158
14.143.2.1 mManufacturerID	1159
14.143.2.2 mProductID	1159
14.143.2.3 mPluginID	1159
14.144 AAX_StLock_Guard Class Reference	1159
14.144.1 Description	1159
14.144.2 Constructor & Destructor Documentation	1160
14.144.2.1 AAX_StLock_Guard()	1160
14.144.2.2 ~AAX_StLock_Guard()	1160
14.145 AAX_TransportStateInfo_V1 Struct Reference	1161
14.145.1 Description	1161
14.145.2 Constructor & Destructor Documentation	1161
14.145.2.1 AAX_TransportStateInfo_V1()	1161
14.145.3 Member Function Documentation	1162
14.145.3.1 ToString()	1162
14.145.4 Member Data Documentation	1162
14.145.4.1 mTransportState	1162
14.145.4.2 mRecordMode	1162
14.145.4.3 mIsRecordEnabled	1163
14.145.4.4 mIsRecording	1163
14.145.4.5 mIsLoopEnabled	1163
14.146 AAX_VAutomationDelegate Class Reference	1163
14.146.1 Description	1164
14.146.2 Constructor & Destructor Documentation	1164

14.146.2.1 AAX_VAutomationDelegate()	1165
14.146.2.2 ~AAX_VAutomationDelegate()	1165
14.146.3 Member Function Documentation	1165
14.146.3.1 GetUnknown()	1165
14.146.3.2 RegisterParameter()	1165
14.146.3.3 UnregisterParameter()	1165
14.146.3.4 PostSetValueRequest()	1166
14.146.3.5 PostCurrentValue()	1166
14.146.3.6 PostTouchRequest()	1167
14.146.3.7 PostReleaseRequest()	1167
14.146.3.8 GetTouchState()	1167
14.146.3.9 ParameterNameChanged()	1168
14.147 AAX_VCollection Class Reference	1168
14.147.1 Description	1169
14.147.2 Constructor & Destructor Documentation	1170
14.147.2.1 AAX_VCollection()	1170
14.147.2.2 ~AAX_VCollection()	1170
14.147.3 Member Function Documentation	1170
14.147.3.1 NewDescriptor()	1170
14.147.3.2 AddEffect()	1171
14.147.3.3 SetManufacturerName()	1171
14.147.3.4 AddPackageName()	1171
14.147.3.5 SetPackageVersion()	1173
14.147.3.6 NewPropertyMap()	1173
14.147.3.7 SetProperties()	1173
14.147.3.8 DescriptionHost() [1/2]	1174
14.147.3.9 DescriptionHost() [2/2]	1174
14.147.3.10 HostDefinition()	1174
14.147.3.11 GetIUnknown()	1175
14.148 AAX_VComponentDescriptor Class Reference	1175
14.148.1 Description	1175
14.148.2 Constructor & Destructor Documentation	1178
14.148.2.1 AAX_VComponentDescriptor()	1178
14.148.2.2 ~AAX_VComponentDescriptor()	1178
14.148.3 Member Function Documentation	1178
14.148.3.1 Clear()	1178
14.148.3.2 AddReservedField()	1178
14.148.3.3 AddAudioIn()	1179
14.148.3.4 AddAudioOut()	1179
14.148.3.5 AddAudioBufferLength()	1180
14.148.3.6 AddSampleRate()	1180
14.148.3.7 AddClock()	1180

14.148.3.8	AddSideChainIn()	1181
14.148.3.9	AddDataInPort()	1181
14.148.3.10	AddAuxOutputStem()	1182
14.148.3.11	AddPrivateData()	1183
14.148.3.12	AddTemporaryData()	1183
14.148.3.13	AddDmaInstance()	1184
14.148.3.14	AddMeters()	1184
14.148.3.15	AddMIDINode()	1186
14.148.3.16	NewPropertyMap()	1186
14.148.3.17	DuplicatePropertyMap()	1187
14.148.3.18	AddProcessProc_Native()	1187
14.148.3.19	AddProcessProc_TI()	1188
14.148.3.20	AddProcessProc()	1188
14.148.3.21	GetUnknown()	1190
14.148.4	Friends And Related Function Documentation	1190
14.148.4.1	AAX_VPropertyMap	1190
14.149	AAX_VController Class Reference	1190
14.149.1	Description	1191
14.149.2	Constructor & Destructor Documentation	1192
14.149.2.1	AAX_VController()	1192
14.149.2.2	~AAX_VController()	1192
14.149.3	Member Function Documentation	1193
14.149.3.1	GetEffectID()	1193
14.149.3.2	GetSampleRate()	1193
14.149.3.3	GetInputStemFormat()	1193
14.149.3.4	GetOutputStemFormat()	1193
14.149.3.5	GetSignalLatency()	1194
14.149.3.6	GetHybridSignalLatency()	1194
14.149.3.7	GetPlugInTargetPlatform()	1195
14.149.3.8	GetIsAudioSuite()	1195
14.149.3.9	GetCycleCount()	1195
14.149.3.10	GetTODLocation()	1196
14.149.3.11	GetCurrentAutomationTimestamp()	1197
14.149.3.12	GetHostName()	1197
14.149.3.13	SetSignalLatency()	1197
14.149.3.14	SetCycleCount()	1198
14.149.3.15	PostPacket()	1199
14.149.3.16	SendNotification() [1/2]	1200
14.149.3.17	SendNotification() [2/2]	1200
14.149.3.18	GetCurrentMeterValue()	1201
14.149.3.19	GetMeterPeakValue()	1201
14.149.3.20	ClearMeterPeakValue()	1201

14.149.3.21	GetMeterClipped()	1202
14.149.3.22	ClearMeterClipped()	1202
14.149.3.23	GetMeterCount()	1202
14.149.3.24	GetNextMIDIPacket()	1204
14.149.3.25	CreateTableCopyForEffect()	1204
14.149.3.26	CreateTableCopyForLayout()	1205
14.149.3.27	CreateTableCopyForEffectFromFile()	1205
14.149.3.28	CreateTableCopyForLayoutFromFile()	1206
14.150	AAX_VDataBufferWrapper Class Reference	1207
14.150.1	Description	1208
14.150.2	Constructor & Destructor Documentation	1208
14.150.2.1	AAX_VDataBufferWrapper()	1208
14.150.2.2	~AAX_VDataBufferWrapper()	1208
14.150.3	Member Function Documentation	1208
14.150.3.1	Type()	1209
14.150.3.2	Size()	1209
14.150.3.3	Data()	1209
14.151	AAX_VDescriptionHost Class Reference	1210
14.151.1	Description	1210
14.151.2	Constructor & Destructor Documentation	1211
14.151.2.1	AAX_VDescriptionHost()	1211
14.151.2.2	~AAX_VDescriptionHost()	1211
14.151.3	Member Function Documentation	1211
14.151.3.1	AcquireFeatureProperties()	1211
14.151.3.2	Supported()	1212
14.151.3.3	DescriptionHost() [1/2]	1212
14.151.3.4	DescriptionHost() [2/2]	1212
14.151.3.5	HostDefinition()	1212
14.152	AAX_VEffectDescriptor Class Reference	1213
14.152.1	Description	1213
14.152.2	Constructor & Destructor Documentation	1214
14.152.2.1	AAX_VEffectDescriptor()	1215
14.152.2.2	~AAX_VEffectDescriptor()	1215
14.152.3	Member Function Documentation	1215
14.152.3.1	NewComponentDescriptor()	1215
14.152.3.2	AddComponent()	1215
14.152.3.3	AddName()	1216
14.152.3.4	AddCategory()	1216
14.152.3.5	AddCategoryBypassParameter()	1216
14.152.3.6	AddProcPtr()	1217
14.152.3.7	NewPropertyMap()	1217
14.152.3.8	SetProperties()	1217

14.152.3.9 AddResourceInfo()	1218
14.152.3.10 AddMeterDescription()	1218
14.152.3.11 AddControlMIDINode()	1218
14.152.3.12 GetUnknown()	1219
14.153 AAX_VFeatureInfo Class Reference	1219
14.153.1 Description	1220
14.153.2 Constructor & Destructor Documentation	1220
14.153.2.1 AAX_VFeatureInfo()	1220
14.153.2.2 ~AAX_VFeatureInfo()	1220
14.153.3 Member Function Documentation	1220
14.153.3.1 SupportLevel()	1221
14.153.3.2 AcquireProperties()	1221
14.153.3.3 ID()	1221
14.154 AAX_VHostProcessorDelegate Class Reference	1222
14.154.1 Description	1222
14.154.2 Constructor & Destructor Documentation	1223
14.154.2.1 AAX_VHostProcessorDelegate()	1223
14.154.3 Member Function Documentation	1223
14.154.3.1 GetAudio()	1223
14.154.3.2 GetSideChainInputNum()	1224
14.154.3.3 ForceAnalyze()	1224
14.154.3.4 ForceProcess()	1225
14.155 AAX_VHostServices Class Reference	1225
14.155.1 Description	1226
14.155.2 Constructor & Destructor Documentation	1226
14.155.2.1 AAX_VHostServices()	1226
14.155.2.2 ~AAX_VHostServices()	1226
14.155.3 Member Function Documentation	1226
14.155.3.1 HandleAssertFailure()	1227
14.155.3.2 Trace()	1227
14.155.3.3 StackTrace()	1227
14.156 AAX_VPageTable Class Reference	1228
14.156.1 Description	1229
14.156.2 Constructor & Destructor Documentation	1231
14.156.2.1 AAX_VPageTable()	1231
14.156.2.2 ~AAX_VPageTable()	1231
14.156.3 Member Function Documentation	1231
14.156.3.1 Clear()	1231
14.156.3.2 Empty()	1231
14.156.3.3 GetNumPages()	1232
14.156.3.4 InsertPage()	1232
14.156.3.5 RemovePage()	1232

14.156.3.6	GetNumMappedParameterIDs()	1233
14.156.3.7	ClearMappedParameter()	1233
14.156.3.8	GetMappedParameterID()	1234
14.156.3.9	MapParameterID()	1234
14.156.3.10	GetNumParametersWithNameVariations()	1235
14.156.3.11	GetNameVariationParameterIDAtIndex()	1235
14.156.3.12	GetNumNameVariationsForParameter()	1236
14.156.3.13	GetParameterNameVariationAtIndex()	1236
14.156.3.14	GetParameterNameVariationOfLength()	1237
14.156.3.15	ClearParameterNameVariations()	1238
14.156.3.16	ClearNameVariationsForParameter()	1239
14.156.3.17	SetParameterNameVariation()	1239
14.156.3.18	AsUnknown() [1/2]	1240
14.156.3.19	AsUnknown() [2/2]	1240
14.156.3.20	IsSupported()	1240
14.157	AAX_VPrivateDataAccess Class Reference	1241
14.157.1	Description	1241
14.157.2	Constructor & Destructor Documentation	1242
14.157.2.1	AAX_VPrivateDataAccess()	1242
14.157.2.2	~AAX_VPrivateDataAccess()	1242
14.157.3	Member Function Documentation	1242
14.157.3.1	ReadPortDirect()	1242
14.157.3.2	WritePortDirect()	1243
14.158	AAX_VPropertyMap Class Reference	1243
14.158.1	Description	1244
14.158.2	Constructor & Destructor Documentation	1245
14.158.2.1	~AAX_VPropertyMap()	1245
14.158.3	Member Function Documentation	1245
14.158.3.1	Create()	1246
14.158.3.2	Acquire()	1246
14.158.3.3	GetProperty()	1246
14.158.3.4	GetPointerProperty()	1246
14.158.3.5	AddProperty()	1247
14.158.3.6	AddPointerProperty() [1/2]	1247
14.158.3.7	AddPointerProperty() [2/2]	1248
14.158.3.8	RemoveProperty()	1248
14.158.3.9	AddPropertyWithIDArray()	1249
14.158.3.10	GetPropertyWithIDArray()	1249
14.158.3.11	GetIUnknown()	1249
14.159	AAX_VSessionDocument Class Reference	1250
14.159.1	Constructor & Destructor Documentation	1251
14.159.1.1	AAX_VSessionDocument()	1251

14.159.1.2 ~AAX_VSessionDocument()	1251
14.159.2 Member Function Documentation	1251
14.159.2.1 Clear()	1251
14.159.2.2 Valid()	1251
14.159.2.3 GetTempoMap()	1252
14.159.2.4 GetDocumentData()	1252
14.160 AAX_VTask Class Reference	1252
14.160.1 Description	1253
14.160.2 Constructor & Destructor Documentation	1254
14.160.2.1 AAX_VTask()	1254
14.160.2.2 ~AAX_VTask()	1254
14.160.3 Member Function Documentation	1254
14.160.3.1 GetType()	1254
14.160.3.2 GetArgumentOfType()	1255
14.160.3.3 SetProgress()	1255
14.160.3.4 GetProgress()	1255
14.160.3.5 AddResult()	1256
14.160.3.6 SetDone()	1256
14.161 AAX_VTransport Class Reference	1256
14.161.1 Description	1257
14.161.2 Constructor & Destructor Documentation	1259
14.161.2.1 AAX_VTransport()	1259
14.161.2.2 ~AAX_VTransport()	1259
14.161.3 Member Function Documentation	1259
14.161.3.1 GetCurrentTempo()	1259
14.161.3.2 GetCurrentMeter()	1260
14.161.3.3 IsTransportPlaying()	1260
14.161.3.4 GetCurrentTickPosition()	1260
14.161.3.5 GetCurrentLoopPosition()	1261
14.161.3.6 GetCurrentNativeSampleLocation()	1261
14.161.3.7 GetCustomTickPosition()	1262
14.161.3.8 GetBarBeatPosition()	1262
14.161.3.9 GetTicksPerQuarter()	1263
14.161.3.10 GetCurrentTicksPerBeat()	1263
14.161.3.11 GetTimelineSelectionStartPosition()	1263
14.161.3.12 GetTimeCodeInfo()	1264
14.161.3.13 GetFeetFramesInfo()	1264
14.161.3.14 IsMetronomeEnabled()	1265
14.161.3.15 GetHDTTimeCodeInfo()	1265
14.161.3.16 GetTimelineSelectionEndPosition()	1265
14.161.3.17 RequestTransportStart()	1266
14.161.3.18 RequestTransportStop()	1266

14.162 AAX_VViewContainer Class Reference	1267
14.162.1 Description	1267
14.162.2 Constructor & Destructor Documentation	1268
14.162.2.1 AAX_VViewContainer()	1268
14.162.2.2 ~AAX_VViewContainer()	1268
14.162.3 Member Function Documentation	1268
14.162.3.1 GetType()	1269
14.162.3.2 GetPtr()	1269
14.162.3.3 GetModifiers()	1269
14.162.3.4 SetViewSize()	1269
14.162.3.5 HandleParameterMouseDown()	1270
14.162.3.6 HandleParameterMouseDrag()	1270
14.162.3.7 HandleParameterMouseUp()	1271
14.162.3.8 HandleParameterMouseEnter()	1271
14.162.3.9 HandleParameterMouseExit()	1271
14.162.3.10 HandleMultipleParametersMouseDown()	1272
14.162.3.11 HandleMultipleParametersMouseDrag()	1272
14.162.3.12 HandleMultipleParametersMouseUp()	1273
14.163 AAX::Exception::Any Class Reference	1273
14.163.1 Description	1274
14.163.2 Constructor & Destructor Documentation	1274
14.163.2.1 ~Any()	1275
14.163.2.2 Any() [1/2]	1275
14.163.2.3 Any() [2/2]	1275
14.163.3 Member Function Documentation	1275
14.163.3.1 operator=()	1275
14.163.3.2 AAX_DEFAULT_MOVE_CTOR()	1275
14.163.3.3 AAX_DEFAULT_MOVE_OPER()	1275
14.163.3.4 What()	1276
14.163.3.5 Desc()	1276
14.163.3.6 Function()	1276
14.163.3.7 Line()	1276
14.164 AAX_CChunkDataParser::DataValue Struct Reference	1277
14.164.1 Constructor & Destructor Documentation	1277
14.164.1.1 DataValue()	1278
14.164.2 Member Data Documentation	1278
14.164.2.1 mDataType	1278
14.164.2.2 mDataName	1278
14.164.2.3 mIntValue	1278
14.164.2.4 mStringValue	1278
14.165 IACFDefinition Interface Reference	1279
14.165.1 Description	1279

14.165.2 Member Function Documentation	1280
14.165.2.1 DefineAttribute()	1280
14.165.2.2 GetAttributeInfo()	1281
14.165.2.3 CopyAttribute()	1281
14.166 IACFUnknown Interface Reference	1282
14.166.1 Description	1282
14.166.2 Member Function Documentation	1283
14.166.2.1 QueryInterface()	1283
14.166.2.2 AddRef()	1284
14.166.2.3 Release()	1284
14.167 AAX::Exception::ResultError Class Reference	1284
14.167.1 Description	1285
14.167.2 Constructor & Destructor Documentation	1286
14.167.2.1 ResultError() [1/3]	1286
14.167.2.2 ResultError() [2/3]	1286
14.167.2.3 ResultError() [3/3]	1286
14.167.3 Member Function Documentation	1286
14.167.3.1 FormatResult()	1286
14.167.3.2 Result()	1287
14.168 SAutoArray< T > Struct Template Reference	1287
14.168.1 Constructor & Destructor Documentation	1287
14.168.1.1 SAutoArray()	1287
14.168.1.2 ~SAutoArray()	1287
14.168.2 Member Function Documentation	1288
14.168.2.1 Reset()	1288
14.168.2.2 Get()	1288
14.169 AAX_I_SessionDocument::TempoMap Class Reference	1288
14.169.1 Constructor & Destructor Documentation	1289
14.169.1.1 ~TempoMap()	1289
14.169.2 Member Function Documentation	1289
14.169.2.1 Size()	1289
14.169.2.2 Data()	1289
14.170 AAX_V_SessionDocument::VTempoMap Class Reference	1290
14.170.1 Constructor & Destructor Documentation	1291
14.170.1.1 ~VTempoMap()	1291
14.170.1.2 VTempoMap()	1291
14.170.2 Member Function Documentation	1291
14.170.2.1 Size()	1291
14.170.2.2 Data()	1291
15 File Documentation	1293
15.1 AAX_ACFInterface.doxygen File Reference	1293

15.1.1 Typedef Documentation	1293
15.1.1.1 acfUID	1293
15.1.1.2 acfIID	1294
15.2 AAX_AdditionalFeatures_Algorithm.doxygen File Reference	1294
15.3 AAX_AdditionalFeatures_AOSandSidechain.doxygen File Reference	1294
15.4 AAX_AdditionalFeatures_CurveDisplays.doxygen File Reference	1294
15.5 AAX_AdditionalFeatures_Hybrid.doxygen File Reference	1294
15.6 AAX_AdditionalFeatures_Meters.doxygen File Reference	1294
15.7 AAX_AdditionalFeatures_MIDI.doxygen File Reference	1294
15.8 AAX_AuxInterface_DirectData.doxygen File Reference	1294
15.9 AAX_AuxInterface_HostProcessor.doxygen File Reference	1294
15.10 AAX_AuxInterface_TaskAgent.doxygen File Reference	1294
15.11 AAX_BugList.doxygen File Reference	1294
15.12 AAX_CommonInterface_Algorithm.doxygen File Reference	1294
15.13 AAX_CommonInterface_Communication.doxygen File Reference	1294
15.14 AAX_CommonInterface_DataModel.doxygen File Reference	1294
15.15 AAX_CommonInterface_Describe.doxygen File Reference	1294
15.16 AAX_CommonInterface_FormatSpecification.doxygen File Reference	1295
15.17 AAX_CommonInterface_GUI.doxygen File Reference	1295
15.18 AAX_DigiTrace_Guide.doxygen File Reference	1295
15.19 AAX_DistributingYourPlugIn.doxygen File Reference	1295
15.20 AAX_DocsDirectory.doxygen File Reference	1295
15.21 AAX_Getting_Started_Guide.doxygen File Reference	1295
15.22 AAX_HostSupport.doxygen File Reference	1295
15.23 AAX_InstrumentParameters.doxygen File Reference	1295
15.24 AAX_InterfaceList.doxygen File Reference	1295
15.25 AAX_LinkedParameters.doxygen File Reference	1295
15.26 AAX_Media_Composer_Guide.doxygen File Reference	1295
15.27 AAX_OtherExtensions.doxygen File Reference	1295
15.28 AAX_Page_Table_Guide.doxygen File Reference	1295
15.29 AAX_ParameterAutomation.doxygen File Reference	1295
15.30 AAX_ParameterManager.doxygen File Reference	1295
15.31 AAX_ParameterUpdateProtocol.doxygen File Reference	1295
15.32 AAX_ParameterUpdateTiming.doxygen File Reference	1295
15.33 AAX_Pro_Tools_Guide.doxygen File Reference	1296
15.34 AAX_RealTimePerformance.doxygen File Reference	1296
15.35 AAX_RelatedTypes.doxygen File Reference	1296
15.36 AAX_SDK_ChangeLog.doxygen File Reference	1296
15.37 AAX_SDK_ExamplePlugIns.doxygen File Reference	1296
15.38 AAX_SDK_GUIExtensions.doxygen File Reference	1296
15.39 AAX_TI_Guide.doxygen File Reference	1296
15.40 AAX_Troubleshooting.doxygen File Reference	1296

15.41 AAX_VENUE_Guide.doxygen File Reference	1296
15.42 DSH_Guide.doxygen File Reference	1296
15.43 DTT_Guide.doxygen File Reference	1296
15.44 ReadMe.doxygen File Reference	1296
15.45 AAX_MIDILogging.cpp File Reference	1296
15.46 AAX_MIDILogging.h File Reference	1297
15.46.1 Description	1297
15.47 AAX_MIDILogging.h	1297
15.48 AAX_PluginBundleLocation.h File Reference	1297
15.48.1 Description	1297
15.49 AAX_PluginBundleLocation.h	1298
15.50 AAX_CMonolithicParameters.cpp File Reference	1298
15.51 AAX_CMonolithicParameters.h File Reference	1298
15.51.1 Description	1299
15.51.2 Macro Definition Documentation	1299
15.51.2.1 kMaxAdditionalMIDINodes	1299
15.51.2.2 kMaxAuxOutputStems	1299
15.51.2.3 kSynchronizedParameterQueueSize	1299
15.52 AAX_CMonolithicParameters.h	1300
15.53 AAX.h File Reference	1303
15.53.1 Description	1303
15.53.2 Macro Definition Documentation	1306
15.53.2.1 TI_VERSION	1306
15.53.2.2 AAX_CPP11_SUPPORT	1306
15.53.2.3 AAX_OVERRIDE	1307
15.53.2.4 AAX_FINAL	1307
15.53.2.5 AAX_DEFAULT_DTOR	1307
15.53.2.6 AAX_DEFAULT_DTOR_OVERRIDE	1307
15.53.2.7 AAX_DEFAULT_CTOR	1307
15.53.2.8 AAX_DEFAULT_COPY_CTOR	1307
15.53.2.9 AAX_DEFAULT_MOVE_CTOR	1308
15.53.2.10 AAX_DEFAULT_ASGN_OPER	1308
15.53.2.11 AAX_DEFAULT_MOVE_OPER	1308
15.53.2.12 AAX_DELETE	1308
15.53.2.13 AAX_CONSTEXPR	1308
15.53.2.14 AAX_UNIQUE_PTR	1309
15.53.2.15 AAXPointer_32bit	1309
15.53.2.16 AAXPointer_64bit	1309
15.53.2.17 AAX_PointerSize	1309
15.53.2.18 AAX_ALIGN_FILE_HOST	1309
15.53.2.19 AAX_ALIGN_FILE_ALG	1310
15.53.2.20 AAX_ALIGN_FILE_RESET	1310

15.53.2.21 AAX_ALIGN_FILE_BEGIN	1310
15.53.2.22 AAX_ALIGN_FILE_END	1310
15.53.2.23 AAX_CALLBACK	1311
15.53.2.24 AAX_PREPROCESSOR_CONCAT_HELPER	1311
15.53.2.25 AAX_PREPROCESSOR_CONCAT	1311
15.53.2.26 AAX_FIELD_INDEX	1311
15.53.3 Typedef Documentation	1311
15.53.3.1 AAX_CIndex	1311
15.53.3.2 AAX_CCount	1312
15.53.3.3 AAX_CBoolean	1312
15.53.3.4 AAX_CSelector	1312
15.53.3.5 AAX_CTimestamp	1312
15.53.3.6 AAX_CTimeOfDay	1312
15.53.3.7 AAX_CTransportCounter	1313
15.53.3.8 AAX_CSampleRate	1313
15.53.3.9 AAX_CTypeID	1313
15.53.3.10 AAX_Result	1313
15.53.3.11 AAX_CPropertyValue	1313
15.53.3.12 AAX_CPropertyValue64	1313
15.53.3.13 AAX_CPointerPropertyValue	1314
15.53.3.14 AAX_CTargetPlatform	1314
15.53.3.15 AAX_CFieldIndex	1314
15.53.3.16 AAX_CComponentID	1314
15.53.3.17 AAX_CMeterID	1314
15.53.3.18 AAX_CParamID	1315
15.53.3.19 AAX_CPageTableParamID	1315
15.53.3.20 AAX_CEffectID	1315
15.53.3.21 acfUID	1315
15.53.3.22 AAX_Feature_UID	1316
15.53.3.23 AAX_CAudioInPort	1316
15.53.3.24 AAX_CAudioOutPort	1316
15.53.3.25 AAX_CMeterPort	1316
15.53.3.26 AAX_SPlugInChunkHeader	1317
15.53.3.27 AAX_SPlugInChunk	1317
15.53.3.28 AAX_SPlugInChunkPtr	1317
15.53.3.29 AAX_SPlugInIdentifierTriad	1317
15.53.3.30 AAX_SPlugInIdentifierTriadPtr	1317
15.53.4 Function Documentation	1317
15.53.4.1 sampleRateInMask()	1317
15.53.4.2 getLowestSampleRateInMask()	1318
15.53.4.3 getMaskForSampleRate()	1318
15.53.5 Variable Documentation	1318

15.53.5.1 kAAX_ParameterIdentifierMaxSize	1318
15.54 AAX.h	1319
15.55 AAX_Assert.h File Reference	1323
15.55.1 Description	1323
15.55.2 Macro Definition Documentation	1325
15.55.2.1 kAAX_Trace_Priority_None	1325
15.55.2.2 kAAX_Trace_Priority_Critical	1325
15.55.2.3 kAAX_Trace_Priority_High	1325
15.55.2.4 kAAX_Trace_Priority_Normal	1325
15.55.2.5 kAAX_Trace_Priority_Low	1325
15.55.2.6 kAAX_Trace_Priority_Lowest	1326
15.55.2.7 AAX_TRACE_RELEASE	1326
15.55.2.8 AAX_STACKTRACE_RELEASE	1326
15.55.2.9 AAX_TRACEORSTACKTRACE_RELEASE	1327
15.55.2.10 AAX_ASSERT	1327
15.55.2.11 AAX_DEBUGASSERT	1328
15.55.2.12 AAX_TRACE	1328
15.55.2.13 AAX_STACKTRACE	1328
15.55.2.14 AAX_TRACEORSTACKTRACE	1328
15.55.3 Typedef Documentation	1329
15.55.3.1 AAX_ETracePriority	1329
15.56 AAX_Assert.h	1329
15.57 AAX_Atomic.h File Reference	1331
15.57.1 Description	1331
15.57.2 Macro Definition Documentation	1332
15.57.2.1 AAX_ATOMIC_H_	1332
15.57.3 Function Documentation	1332
15.57.3.1 AAX_Atomic_IncThenGet_32()	1332
15.57.3.2 AAX_Atomic_DecThenGet_32()	1332
15.57.3.3 AAX_Atomic_Exchange_32()	1332
15.57.3.4 AAX_Atomic_Exchange_64()	1333
15.57.3.5 AAX_Atomic_Exchange_Pointer()	1333
15.57.3.6 AAX_Atomic_CompareAndExchange_32()	1334
15.57.3.7 AAX_Atomic_CompareAndExchange_64()	1334
15.57.3.8 AAX_Atomic_CompareAndExchange_Pointer()	1335
15.57.3.9 AAX_Atomic_Load_Pointer()	1335
15.58 AAX_Atomic.h	1335
15.59 AAX_Callbacks.h File Reference	1339
15.59.1 Description	1339
15.59.2 Typedef Documentation	1339
15.59.2.1 AAXCreateObjectProc	1339
15.59.2.2 AAX_CProcessProc	1340

15.59.2.3 AAX_CPacketAllocator	1340
15.59.2.4 AAX_CInstanceInitProc	1340
15.59.2.5 AAX_CBackgroundProc	1341
15.59.2.6 AAX_CInitPrivateDataProc	1341
15.59.3 Enumeration Type Documentation	1342
15.59.3.1 AAX_CProcPtrID	1342
15.60 AAX_Callbacks.h	1342
15.61 AAX_CArrayDataBuffer.h File Reference	1343
15.61.1 Macro Definition Documentation	1344
15.61.1.1 AAX_CArrayDataBuffer_H	1344
15.62 AAX_CArrayDataBuffer.h	1344
15.63 AAX_CAtomicQueue.h File Reference	1346
15.63.1 Description	1346
15.64 AAX_CAtomicQueue.h	1346
15.65 AAX_CAutoreleasePool.h File Reference	1350
15.65.1 Description	1350
15.65.2 Macro Definition Documentation	1350
15.65.2.1 _AAX_CAUTORELEASEPOOL_H	1350
15.66 AAX_CAutoreleasePool.h	1350
15.67 AAX_CBinaryDisplayDelegate.h File Reference	1351
15.67.1 Description	1351
15.68 AAX_CBinaryDisplayDelegate.h	1351
15.69 AAX_CBinaryTaperDelegate.h File Reference	1354
15.69.1 Description	1354
15.70 AAX_CBinaryTaperDelegate.h	1354
15.71 AAX_CChunkDataParser.h File Reference	1355
15.71.1 Description	1355
15.71.2 Macro Definition Documentation	1356
15.71.2.1 AAX_CHUNKDATAPARSER_H	1356
15.72 AAX_CChunkDataParser.h	1357
15.73 AAX_CDecibelDisplayDelegateDecorator.h File Reference	1358
15.73.1 Description	1358
15.74 AAX_CDecibelDisplayDelegateDecorator.h	1358
15.75 AAX_CEffectDirectData.h File Reference	1360
15.75.1 Description	1360
15.75.2 Macro Definition Documentation	1361
15.75.2.1 AAX_CEFFECTDIRECTDATA_H	1361
15.76 AAX_CEffectDirectData.h	1361
15.77 AAX_CEffectGUI.h File Reference	1362
15.77.1 Description	1362
15.78 AAX_CEffectGUI.h	1362
15.79 AAX_CEffectParameters.h File Reference	1363

15.79.1 Description	1364
15.79.2 Function Documentation	1364
15.79.2.1 NormalizedToInt32()	1364
15.79.2.2 Int32ToNormalized()	1364
15.79.2.3 BoolToNormalized()	1364
15.79.3 Variable Documentation	1365
15.79.3.1 cPreviewID	1365
15.79.3.2 cDefaultMasterBypassID	1365
15.80 AAX_CEffectParameters.h	1365
15.81 AAX_CHostProcessor.h File Reference	1367
15.81.1 Description	1367
15.82 AAX_CHostProcessor.h	1368
15.83 AAX_CHostServices.h File Reference	1369
15.83.1 Description	1369
15.84 AAX_CHostServices.h	1369
15.85 AAX_CLinearTaperDelegate.h File Reference	1370
15.85.1 Description	1370
15.86 AAX_CLinearTaperDelegate.h	1370
15.87 AAX_CLogTaperDelegate.h File Reference	1372
15.87.1 Description	1372
15.88 AAX_CLogTaperDelegate.h	1372
15.89 AAX_CMutex.h File Reference	1373
15.89.1 Description	1374
15.90 AAX_CMutex.h	1374
15.91 AAX_CNumberDisplayDelegate.h File Reference	1374
15.91.1 Description	1375
15.92 AAX_CNumberDisplayDelegate.h	1375
15.93 AAX_CommonConversions.h File Reference	1376
15.93.1 Function Documentation	1377
15.93.1.1 GainToDB()	1377
15.93.1.2 DBToGain()	1377
15.93.1.3 LongToDouble()	1378
15.93.1.4 DoubleToLong()	1378
15.93.1.5 DoubleToDSPCoef()	1378
15.93.1.6 DSPCoefToDouble()	1379
15.93.1.7 ThirtyTwoBitDSPCoefToDouble()	1379
15.93.1.8 DoubleTo32BitDSPCoefRnd()	1380
15.93.1.9 DoubleTo32BitDSPCoef()	1380
15.93.1.10 DoubleToDSPCoefRnd()	1380
15.93.2 Variable Documentation	1380
15.93.2.1 k32BitPosMax	1380
15.93.2.2 k32BitAbsMax	1381

15.93.2.3 k32BitNegMax	1381
15.93.2.4 k56kFracPosMax	1381
15.93.2.5 k56kFracAbsMax	1381
15.93.2.6 k56kFracHalf	1381
15.93.2.7 k56kFracNegOne	1381
15.93.2.8 k56kFracNegMax	1381
15.93.2.9 k56kFracZero	1382
15.93.2.10 kOneOver56kFracAbsMax	1382
15.93.2.11 k56kFloatPosMax	1382
15.93.2.12 k56kFloatNegMax	1382
15.93.2.13 kNeg144DB	1382
15.93.2.14 kNeg144Gain	1382
15.94 AAX_CommonConversions.h	1383
15.95 AAX_CPacketDispatcher.h File Reference	1384
15.95.1 Description	1384
15.96 AAX_CPacketDispatcher.h	1385
15.97 AAX_CParameter.h File Reference	1387
15.97.1 Description	1387
15.98 AAX_CParameter.h	1387
15.99 AAX_CParameterManager.h File Reference	1400
15.99.1 Description	1400
15.100 AAX_CParameterManager.h	1401
15.101 AAX_CPercentDisplayDelegateDecorator.h File Reference	1401
15.101.1 Description	1402
15.101.2 Macro Definition Documentation	1402
15.101.2.1 AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H	1402
15.102 AAX_CPercentDisplayDelegateDecorator.h	1402
15.103 AAX_CPieceWiseLinearTaperDelegate.h File Reference	1403
15.103.1 Description	1404
15.104 AAX_CPieceWiseLinearTaperDelegate.h	1404
15.105 AAX_CRangeTaperDelegate.h File Reference	1407
15.105.1 Description	1407
15.106 AAX_CRangeTaperDelegate.h	1407
15.107 AAX_CSessionDocumentClient.h File Reference	1410
15.107.1 Macro Definition Documentation	1410
15.107.1.1 AAX_CSessionDocumentClient_H	1410
15.108 AAX_CSessionDocumentClient.h	1410
15.109 AAX_CStateDisplayDelegate.h File Reference	1411
15.109.1 Description	1411
15.110 AAX_CStateDisplayDelegate.h	1412
15.111 AAX_CStateTaperDelegate.h File Reference	1414
15.111.1 Description	1415

15.112 AAX_CStateTaperDelegate.h	1415
15.113 AAX_CString.h File Reference	1416
15.113.1 Description	1416
15.113.2 Macro Definition Documentation	1417
15.113.2.1 AAX_CSTRING_H	1417
15.113.3 Function Documentation	1417
15.113.3.1 operator+() [1/3]	1417
15.113.3.2 operator+() [2/3]	1417
15.113.3.3 operator+() [3/3]	1417
15.114 AAX_CString.h	1418
15.115 AAX_CStringDataBuffer.h File Reference	1420
15.115.1 Macro Definition Documentation	1420
15.115.1.1 AAX_CStringDataBuffer_H	1420
15.116 AAX_CStringDataBuffer.h	1421
15.117 AAX_CStringDisplayDelegate.h File Reference	1422
15.117.1 Description	1422
15.118 AAX_CStringDisplayDelegate.h	1422
15.119 AAX_CTaskAgent.h File Reference	1424
15.119.1 Description	1424
15.120 AAX_CTaskAgent.h	1424
15.121 AAX_CUnitDisplayDelegateDecorator.h File Reference	1425
15.121.1 Description	1425
15.122 AAX_CUnitDisplayDelegateDecorator.h	1425
15.123 AAX_CUnitPrefixDisplayDelegateDecorator.h File Reference	1427
15.123.1 Description	1427
15.124 AAX_CUnitPrefixDisplayDelegateDecorator.h	1427
15.125 AAX_EndianSwap.h File Reference	1429
15.125.1 Description	1429
15.125.2 Macro Definition Documentation	1430
15.125.2.1 ENDIANSWAP_H	1430
15.125.3 Function Documentation	1430
15.125.3.1 AAX_EndianSwapInPlace()	1431
15.125.3.2 AAX_EndianSwap()	1431
15.125.3.3 AAX_BigEndianNativeSwapInPlace()	1432
15.125.3.4 AAX_BigEndianNativeSwap()	1432
15.125.3.5 AAX_LittleEndianNativeSwapInPlace()	1433
15.125.3.6 AAX_LittleEndianNativeSwap()	1433
15.125.3.7 AAX_EndianSwapSequenceInPlace()	1434
15.125.3.8 AAX_BigEndianNativeSwapSequenceInPlace()	1434
15.125.3.9 AAX_LittleEndianNativeSwapSequenceInPlace()	1435
15.126 AAX_EndianSwap.h	1435
15.127 AAX_Enums.h File Reference	1437

15.127.1 Description	1437
15.127.2 Macro Definition Documentation	1446
15.127.2.1 AAX_INT32_MIN	1446
15.127.2.2 AAX_INT32_MAX	1446
15.127.2.3 AAX_UINT32_MIN	1446
15.127.2.4 AAX_UINT32_MAX	1446
15.127.2.5 AAX_INT16_MIN	1446
15.127.2.6 AAX_INT16_MAX	1446
15.127.2.7 AAX_UINT16_MIN	1446
15.127.2.8 AAX_UINT16_MAX	1447
15.127.2.9 AAX_ENUM_SIZE_CHECK	1447
15.127.2.10 AAX_STEM_FORMAT	1447
15.127.2.11 AAX_STEM_FORMAT_CHANNEL_COUNT	1447
15.127.2.12 AAX_STEM_FORMAT_INDEX	1447
15.127.3 Typedef Documentation	1447
15.127.3.1 AAX_EParameterType	1447
15.127.3.2 AAX_EParameterOrientation	1448
15.127.4 Enumeration Type Documentation	1448
15.127.4.1 AAX_EHighlightColor	1448
15.127.4.2 AAX_ETracePriorityHost	1448
15.127.4.3 AAX_ETracePriorityDSP	1449
15.127.4.4 AAX_EModifiers	1449
15.127.4.5 AAX_EAudioBufferLength	1449
15.127.4.6 AAX_EAudioBufferLengthDSP	1450
15.127.4.7 AAX_EAudioBufferLengthNative	1451
15.127.4.8 AAX_EMaxAudioSuiteTracks	1451
15.127.4.9 AAX_EStemFormat	1451
15.127.4.10 AAX_EPlugInCategory	1453
15.127.4.11 AAX_EPlugInStrings	1454
15.127.4.12 AAX_EMeterOrientation	1455
15.127.4.13 AAX_EMeterBallisticType	1456
15.127.4.14 AAX_EMeterType	1456
15.127.4.15 AAX_EResourceType	1456
15.127.4.16 AAX_ENotificationEvent	1457
15.127.4.17 AAX_EHostModeBits	1461
15.127.4.18 AAX_EHostMode	1462
15.127.4.19 AAX_EPrivateDataOptions	1462
15.127.4.20 AAX_EConstraintLocationMask	1463
15.127.4.21 AAX_EConstraintTopology	1463
15.127.4.22 AAX_EComponentInstanceInitAction	1464
15.127.4.23 AAX_ESampleRateMask	1464
15.127.4.24 AAX_EParameterType	1464

15.127.4.25 AAX_EParameterOrientationBits	1465
15.127.4.26 AAX_EParameterValueInfoSelector	1465
15.127.4.27 AAX_EEQBandTypes	1466
15.127.4.28 AAX_EEQInCircuitPolarity	1467
15.127.4.29 AAX_EUseAlternateControl	1467
15.127.4.30 AAX_EMIDINodeType	1467
15.127.4.31 AAX_EUpdateSource	1469
15.127.4.32 AAX_EDataInPortType	1469
15.127.4.33 AAX_EFrameRate	1470
15.127.4.34 AAX_EFeetFramesRate	1470
15.127.4.35 AAX_EMidiGlobalNodeSelectors	1471
15.127.4.36 AAX_EPreviewState	1471
15.127.4.37 AAX_EProcessingState	1472
15.127.4.38 AAX_ETargetPlatform	1473
15.127.4.39 AAX_ESupportLevel	1473
15.127.4.40 AAX_EHostLevel	1473
15.127.4.41 AAX_ETextEncoding	1474
15.127.4.42 AAX_EAssertFlags	1474
15.127.4.43 AAX_ETransportState	1475
15.127.4.44 AAX_ERecordMode	1475
15.127.5 Function Documentation	1475
15.127.5.1 AAX_ENUM_SIZE_CHECK() [1/45]	1476
15.127.5.2 AAX_ENUM_SIZE_CHECK() [2/45]	1476
15.127.5.3 AAX_ENUM_SIZE_CHECK() [3/45]	1476
15.127.5.4 AAX_ENUM_SIZE_CHECK() [4/45]	1476
15.127.5.5 AAX_ENUM_SIZE_CHECK() [5/45]	1476
15.127.5.6 AAX_ENUM_SIZE_CHECK() [6/45]	1476
15.127.5.7 AAX_ENUM_SIZE_CHECK() [7/45]	1476
15.127.5.8 AAX_ENUM_SIZE_CHECK() [8/45]	1477
15.127.5.9 AAX_ENUM_SIZE_CHECK() [9/45]	1477
15.127.5.10 AAX_ENUM_SIZE_CHECK() [10/45]	1477
15.127.5.11 AAX_ENUM_SIZE_CHECK() [11/45]	1477
15.127.5.12 AAX_ENUM_SIZE_CHECK() [12/45]	1477
15.127.5.13 AAX_ENUM_SIZE_CHECK() [13/45]	1477
15.127.5.14 AAX_ENUM_SIZE_CHECK() [14/45]	1477
15.127.5.15 AAX_ENUM_SIZE_CHECK() [15/45]	1478
15.127.5.16 AAX_ENUM_SIZE_CHECK() [16/45]	1478
15.127.5.17 AAX_ENUM_SIZE_CHECK() [17/45]	1478
15.127.5.18 AAX_ENUM_SIZE_CHECK() [18/45]	1478
15.127.5.19 AAX_ENUM_SIZE_CHECK() [19/45]	1478
15.127.5.20 AAX_ENUM_SIZE_CHECK() [20/45]	1478
15.127.5.21 AAX_ENUM_SIZE_CHECK() [21/45]	1478

15.127.5.22 AAX_ENUM_SIZE_CHECK() [22/45]	1479
15.127.5.23 AAX_ENUM_SIZE_CHECK() [23/45]	1479
15.127.5.24 AAX_ENUM_SIZE_CHECK() [24/45]	1479
15.127.5.25 AAX_ENUM_SIZE_CHECK() [25/45]	1479
15.127.5.26 AAX_ENUM_SIZE_CHECK() [26/45]	1479
15.127.5.27 AAX_ENUM_SIZE_CHECK() [27/45]	1479
15.127.5.28 AAX_ENUM_SIZE_CHECK() [28/45]	1479
15.127.5.29 AAX_ENUM_SIZE_CHECK() [29/45]	1480
15.127.5.30 AAX_ENUM_SIZE_CHECK() [30/45]	1480
15.127.5.31 AAX_ENUM_SIZE_CHECK() [31/45]	1480
15.127.5.32 AAX_ENUM_SIZE_CHECK() [32/45]	1480
15.127.5.33 AAX_ENUM_SIZE_CHECK() [33/45]	1480
15.127.5.34 AAX_ENUM_SIZE_CHECK() [34/45]	1480
15.127.5.35 AAX_ENUM_SIZE_CHECK() [35/45]	1480
15.127.5.36 AAX_ENUM_SIZE_CHECK() [36/45]	1481
15.127.5.37 AAX_ENUM_SIZE_CHECK() [37/45]	1481
15.127.5.38 AAX_ENUM_SIZE_CHECK() [38/45]	1481
15.127.5.39 AAX_ENUM_SIZE_CHECK() [39/45]	1481
15.127.5.40 AAX_ENUM_SIZE_CHECK() [40/45]	1481
15.127.5.41 AAX_ENUM_SIZE_CHECK() [41/45]	1481
15.127.5.42 AAX_ENUM_SIZE_CHECK() [42/45]	1481
15.127.5.43 AAX_ENUM_SIZE_CHECK() [43/45]	1482
15.127.5.44 AAX_ENUM_SIZE_CHECK() [44/45]	1482
15.127.5.45 AAX_ENUM_SIZE_CHECK() [45/45]	1482
15.128 AAX_Enums.h	1482
15.129 AAX_EnvironmentUtilities.h File Reference	1489
15.129.1 Description	1489
15.130 AAX_EnvironmentUtilities.h	1489
15.131 AAX_Errors.h File Reference	1490
15.131.1 Description	1490
15.131.2 Enumeration Type Documentation	1491
15.131.2.1 AAX_EError	1491
15.131.3 Function Documentation	1493
15.131.3.1 AAX_ENUM_SIZE_CHECK()	1493
15.132 AAX_Errors.h	1493
15.133 AAX_Exception.h File Reference	1503
15.133.1 Description	1503
15.133.2 Macro Definition Documentation	1504
15.133.2.1 AAX_SWALLOW	1504
15.133.2.2 AAX_SWALLOW_MULT	1504
15.133.2.3 AAX_CAPTURE	1505
15.133.2.4 AAX_CAPTURE_MULT	1505

15.134 AAX_Exception.h	1506
15.135 AAX_Exports.cpp File Reference	1510
15.135.1 Macro Definition Documentation	1510
15.135.1.1 AAX_EXPORT	1511
15.135.2 Function Documentation	1511
15.135.2.1 ACFRegisterPlugin()	1511
15.135.2.2 ACFRegisterComponent()	1511
15.135.2.3 ACFGetClassFactory()	1512
15.135.2.4 ACFCanUnloadNow()	1512
15.135.2.5 ACFStartup()	1513
15.135.2.6 ACFShutdown()	1513
15.135.2.7 ACFGetSDKVersion()	1514
15.136 AAX_GUITypes.h File Reference	1514
15.136.1 Description	1514
15.136.2 Typedef Documentation	1515
15.136.2.1 AAX_Point	1515
15.136.2.2 AAX_Rect	1515
15.136.2.3 AAX_EViewContainer_Type	1515
15.136.3 Enumeration Type Documentation	1516
15.136.3.1 AAX_EViewContainer_Type	1516
15.136.4 Function Documentation	1516
15.136.4.1 operator==() [1/2]	1516
15.136.4.2 operator!=() [1/2]	1516
15.136.4.3 operator<()	1517
15.136.4.4 operator<=()	1517
15.136.4.5 operator>()	1517
15.136.4.6 operator>=()	1517
15.136.4.7 operator==() [2/2]	1517
15.136.4.8 operator!=() [2/2]	1518
15.136.4.9 AAX_ENUM_SIZE_CHECK()	1518
15.137 AAX_GUITypes.h	1518
15.138 AAX_IACFAutomationDelegate.h File Reference	1519
15.138.1 Description	1520
15.139 AAX_IACFAutomationDelegate.h	1520
15.140 AAX_IACFCollection.h File Reference	1521
15.140.1 Description	1521
15.141 AAX_IACFCollection.h	1521
15.142 AAX_IACFComponentDescriptor.h File Reference	1522
15.142.1 Description	1522
15.143 AAX_IACFComponentDescriptor.h	1522
15.144 AAX_IACFController.h File Reference	1523
15.144.1 Description	1523

15.145 AAX_IACFController.h	1524
15.146 AAX_IACFDataBuffer.h File Reference	1526
15.146.1 Macro Definition Documentation	1526
15.146.1.1 AAX_IACFDataBuffer_H	1526
15.147 AAX_IACFDataBuffer.h	1527
15.148 AAX_IACFDescriptionHost.h File Reference	1527
15.149 AAX_IACFDescriptionHost.h	1527
15.150 AAX_IACFEffectDescriptor.h File Reference	1528
15.150.1 Description	1528
15.151 AAX_IACFEffectDescriptor.h	1528
15.152 AAX_IACFEffectDirectData.h File Reference	1529
15.152.1 Description	1529
15.153 AAX_IACFEffectDirectData.h	1530
15.154 AAX_IACFEffectGUI.h File Reference	1530
15.154.1 Description	1530
15.155 AAX_IACFEffectGUI.h	1531
15.156 AAX_IACFEffectParameters.h File Reference	1531
15.156.1 Description	1532
15.157 AAX_IACFEffectParameters.h	1532
15.158 AAX_IACFFeatureInfo.h File Reference	1534
15.159 AAX_IACFFeatureInfo.h	1534
15.160 AAX_IACFHostProcessor.h File Reference	1535
15.160.1 Description	1535
15.161 AAX_IACFHostProcessor.h	1535
15.162 AAX_IACFHostProcessorDelegate.h File Reference	1536
15.163 AAX_IACFHostProcessorDelegate.h	1537
15.164 AAX_IACFHostServices.h File Reference	1537
15.165 AAX_IACFHostServices.h	1538
15.166 AAX_IACFPageTable.h File Reference	1538
15.167 AAX_IACFPageTable.h	1539
15.168 AAX_IACFPageTableController.h File Reference	1539
15.169 AAX_IACFPageTableController.h	1540
15.170 AAX_IACFPrivateDataAccess.h File Reference	1541
15.170.1 Description	1541
15.171 AAX_IACFPrivateDataAccess.h	1541
15.172 AAX_IACFPropertyMap.h File Reference	1542
15.172.1 Description	1542
15.173 AAX_IACFPropertyMap.h	1542
15.174 AAX_IACFSessionDocument.h File Reference	1543
15.174.1 Macro Definition Documentation	1543
15.174.1.1 AAX_IACFSessionDocument_H	1543
15.175 AAX_IACFSessionDocument.h	1544

15.176 AAX_IACFSessionDocumentClient.h File Reference	1544
15.176.1 Macro Definition Documentation	1544
15.176.1.1 AAX_IACFSessionDocumentClient_H	1545
15.177 AAX_IACFSessionDocumentClient.h	1545
15.178 AAX_IACFTask.h File Reference	1545
15.178.1 Description	1545
15.178.2 Macro Definition Documentation	1546
15.178.2.1 AAX_IACFTask_H	1546
15.179 AAX_IACFTask.h	1546
15.180 AAX_IACFTaskAgent.h File Reference	1547
15.180.1 Macro Definition Documentation	1547
15.180.1.1 AAX_IACFTaskAgent_H	1547
15.181 AAX_IACFTaskAgent.h	1547
15.182 AAX_IACFTransport.h File Reference	1548
15.182.1 Description	1548
15.183 AAX_IACFTransport.h	1549
15.184 AAX_IACFTransportControl.h File Reference	1550
15.184.1 Description	1550
15.185 AAX_IACFTransportControl.h	1550
15.186 AAX_IACFViewContainer.h File Reference	1551
15.186.1 Description	1551
15.187 AAX_IACFViewContainer.h	1551
15.188 AAX_IAutomationDelegate.h File Reference	1552
15.188.1 Description	1552
15.189 AAX_IAutomationDelegate.h	1552
15.190 AAX_ICollection.h File Reference	1553
15.190.1 Description	1553
15.191 AAX_ICollection.h	1553
15.192 AAX_IComponentDescriptor.h File Reference	1554
15.192.1 Description	1554
15.193 AAX_IComponentDescriptor.h	1555
15.194 AAX_IContainer.h File Reference	1556
15.194.1 Description	1556
15.195 AAX_IContainer.h	1556
15.196 AAX_IController.h File Reference	1557
15.196.1 Description	1557
15.197 AAX_IController.h	1557
15.198 AAX_IDataBuffer.h File Reference	1559
15.198.1 Macro Definition Documentation	1559
15.198.1.1 AAX_IDataBuffer_H	1560
15.199 AAX_IDataBuffer.h	1560
15.200 AAX_IDataBufferWrapper.h File Reference	1560

15.200.1 Macro Definition Documentation	1561
15.200.1.1 AAX_IDATABUFFERWRAPPER_H	1561
15.201 AAX_IDataBufferWrapper.h	1561
15.202 AAX_IDescriptionHost.h File Reference	1561
15.203 AAX_IDescriptionHost.h	1562
15.204 AAX_IDisplayDelegate.h File Reference	1562
15.204.1 Description	1562
15.205 AAX_IDisplayDelegate.h	1562
15.206 AAX_IDisplayDelegateDecorator.h File Reference	1563
15.206.1 Description	1563
15.207 AAX_IDisplayDelegateDecorator.h	1563
15.208 AAX_IDma.h File Reference	1565
15.208.1 Description	1565
15.208.2 Macro Definition Documentation	1565
15.208.2.1 AAX_IDMA_H	1565
15.208.2.2 AAX_DMA_API	1565
15.209 AAX_IDma.h	1566
15.210 AAX_IEffectDescriptor.h File Reference	1567
15.210.1 Description	1567
15.211 AAX_IEffectDescriptor.h	1567
15.212 AAX_IEffectDirectData.h File Reference	1568
15.212.1 Description	1568
15.213 AAX_IEffectDirectData.h	1568
15.214 AAX_IEffectGUI.h File Reference	1569
15.214.1 Description	1569
15.215 AAX_IEffectGUI.h	1569
15.216 AAX_IEffectParameters.h File Reference	1570
15.216.1 Description	1570
15.217 AAX_IEffectParameters.h	1570
15.218 AAX_IFeatureInfo.h File Reference	1570
15.219 AAX_IFeatureInfo.h	1571
15.220 AAX_IHostProcessor.h File Reference	1571
15.220.1 Description	1571
15.221 AAX_IHostProcessor.h	1572
15.222 AAX_IHostProcessorDelegate.h File Reference	1572
15.222.1 Description	1572
15.223 AAX_IHostProcessorDelegate.h	1572
15.224 AAX_IHostServices.h File Reference	1573
15.224.1 Description	1573
15.225 AAX_IHostServices.h	1573
15.226 AAX_IMIDINode.h File Reference	1574
15.226.1 Description	1574

15.227 AAX_IMIDINode.h	1574
15.228 AAX_Init.h File Reference	1575
15.228.1 Description	1575
15.228.2 Function Documentation	1575
15.228.2.1 AAXRegisterComponent()	1575
15.228.2.2 AAXGetClassFactory()	1576
15.228.2.3 AAXCanUnloadNow()	1577
15.228.2.4 AAXStartup()	1577
15.228.2.5 AAXShutdown()	1578
15.228.2.6 AAXGetSDKVersion()	1578
15.229 AAX_Init.h	1579
15.230 AAX_IPageTable.h File Reference	1579
15.231 AAX_IPageTable.h	1579
15.232 AAX_IParameter.h File Reference	1580
15.232.1 Description	1580
15.233 AAX_IParameter.h	1581
15.234 AAX_IPointerQueue.h File Reference	1583
15.234.1 Description	1583
15.235 AAX_IPointerQueue.h	1583
15.236 AAX_IPrivateDataAccess.h File Reference	1584
15.236.1 Description	1584
15.237 AAX_IPrivateDataAccess.h	1584
15.238 AAX_IPropertyMap.h File Reference	1585
15.238.1 Description	1585
15.239 AAX_IPropertyMap.h	1585
15.240 AAX_ISessionDocument.h File Reference	1586
15.240.1 Macro Definition Documentation	1586
15.240.1.1 AAX_ISessionDocument_H	1586
15.241 AAX_ISessionDocument.h	1586
15.242 AAX_ISessionDocumentClient.h File Reference	1587
15.242.1 Macro Definition Documentation	1587
15.242.1.1 AAX_ISessionDocumentClient_H	1587
15.243 AAX_ISessionDocumentClient.h	1588
15.244 AAX_IString.h File Reference	1588
15.244.1 Description	1588
15.245 AAX_IString.h	1589
15.246 AAX_ITaperDelegate.h File Reference	1589
15.246.1 Description	1589
15.247 AAX_ITaperDelegate.h	1589
15.248 AAX_ITask.h File Reference	1590
15.248.1 Macro Definition Documentation	1590
15.248.1.1 AAX_ITask_H	1590

15.249 AAX_ITask.h	1591
15.250 AAX_ITaskAgent.h File Reference	1591
15.251 AAX_ITaskAgent.h	1591
15.252 AAX_ITransport.h File Reference	1592
15.252.1 Description	1592
15.253 AAX_ITransport.h	1593
15.254 AAX_IViewContainer.h File Reference	1593
15.254.1 Description	1594
15.255 AAX_IViewContainer.h	1594
15.256 AAX_MIDIUtilities.h File Reference	1595
15.256.1 Description	1595
15.257 AAX_MIDIUtilities.h	1596
15.258 AAX_PageTableUtilities.h File Reference	1598
15.259 AAX_PageTableUtilities.h	1598
15.260 AAX_PopStructAlignment.h File Reference	1601
15.260.1 Description	1601
15.261 AAX_PopStructAlignment.h	1601
15.262 AAX_PostStructAlignmentHelper.h File Reference	1602
15.262.1 Description	1602
15.263 AAX_PostStructAlignmentHelper.h	1602
15.264 AAX_PreStructAlignmentHelper.h File Reference	1602
15.264.1 Description	1602
15.265 AAX_PreStructAlignmentHelper.h	1603
15.266 AAX_Properties.h File Reference	1603
15.266.1 Description	1603
15.266.2 Enumeration Type Documentation	1605
15.266.2.1 AAX_EProperty	1605
15.266.3 Function Documentation	1623
15.266.3.1 AAX_ENUM_SIZE_CHECK()	1623
15.267 AAX_Properties.h	1623
15.268 AAX_Push2ByteStructAlignment.h File Reference	1626
15.268.1 Description	1626
15.268.2 Usage notes	1626
15.269 AAX_Push2ByteStructAlignment.h	1626
15.270 AAX_Push4ByteStructAlignment.h File Reference	1627
15.270.1 Description	1627
15.270.2 Usage notes	1627
15.271 AAX_Push4ByteStructAlignment.h	1628
15.272 AAX_Push8ByteStructAlignment.h File Reference	1628
15.272.1 Description	1628
15.272.2 Usage notes	1629
15.273 AAX_Push8ByteStructAlignment.h	1629

15.274 AAX_SessionDocumentTypes.h File Reference	1630
15.274.1 Macro Definition Documentation	1630
15.274.1.1 AAX_SessionDocumentTypes_H	1630
15.274.2 Variable Documentation	1630
15.274.2.1 kAAX_DataBufferType_TempoBreakpointArray	1630
15.275 AAX_SessionDocumentTypes.h	1631
15.276 AAX_SliderConversions.h File Reference	1631
15.276.1 Description	1631
15.276.2 Macro Definition Documentation	1632
15.276.2.1 AAX_SLIDERCONVERSIONS_H	1632
15.276.2.2 AAX_LIMIT	1632
15.276.3 Function Documentation	1632
15.276.3.1 LongControlToNewRange()	1633
15.276.3.2 LongToLongControl()	1633
15.276.3.3 LongControlToDouble()	1633
15.276.3.4 DoubleToLongControl()	1633
15.276.3.5 DoubleToLongControlNonlinear()	1633
15.276.3.6 LongControlToDoubleNonlinear()	1634
15.276.3.7 LongControlToLogDouble()	1634
15.276.3.8 LogDoubleToLongControl()	1634
15.277 AAX_SliderConversions.h	1635
15.278 AAX_StringUtilities.h File Reference	1635
15.278.1 Description	1635
15.278.2 Macro Definition Documentation	1636
15.278.2.1 AAXLibrary_AAX_StringUtilities_h	1636
15.279 AAX_StringUtilities.h	1636
15.280 AAX_StringUtilities.hpp File Reference	1637
15.280.1 Macro Definition Documentation	1638
15.280.1.1 DEFINE_AAX_ERROR_STRING	1638
15.281 AAX_StringUtilities.hpp	1638
15.282 AAX_TransportTypes.h File Reference	1649
15.282.1 Description	1649
15.282.2 Function Documentation	1649
15.282.2.1 operator==()	1649
15.282.2.2 operator!=()	1650
15.283 AAX_TransportTypes.h	1650
15.284 AAX_UIDs.h File Reference	1651
15.284.1 Description	1651
15.284.2 Typedef Documentation	1654
15.284.2.1 AAX_Feature_UID	1654
15.284.2.2 AAX_DocumentData_UID	1654
15.284.3 Variable Documentation	1655

15.284.3.1 AAXCompID_HostServices	1655
15.284.3.2 IID_IAAXHostServicesV1	1655
15.284.3.3 IID_IAAXHostServicesV2	1655
15.284.3.4 IID_IAAXHostServicesV3	1655
15.284.3.5 AAXCompID_AAXCollection	1655
15.284.3.6 IID_IAAXCollectionV1	1656
15.284.3.7 AAXCompID_AAXEffectDescriptor	1656
15.284.3.8 IID_IAAXEffectDescriptorV1	1656
15.284.3.9 IID_IAAXEffectDescriptorV2	1656
15.284.3.10 AAXCompID_AAXComponentDescriptor	1656
15.284.3.11 IID_IAAXComponentDescriptorV1	1656
15.284.3.12 IID_IAAXComponentDescriptorV2	1657
15.284.3.13 IID_IAAXComponentDescriptorV3	1657
15.284.3.14 AAXCompID_AAXPropertyMap	1657
15.284.3.15 IID_IAAXPropertyMapV1	1657
15.284.3.16 IID_IAAXPropertyMapV2	1657
15.284.3.17 IID_IAAXPropertyMapV3	1657
15.284.3.18 AAXCompID_HostProcessorDelegate	1658
15.284.3.19 IID_IAAXHostProcessorDelegateV1	1658
15.284.3.20 IID_IAAXHostProcessorDelegateV2	1658
15.284.3.21 IID_IAAXHostProcessorDelegateV3	1658
15.284.3.22 AAXCompID_AutomationDelegate	1658
15.284.3.23 IID_IAAXAutomationDelegateV1	1658
15.284.3.24 AAXCompID_Controller	1659
15.284.3.25 IID_IAAXControllerV1	1659
15.284.3.26 IID_IAAXControllerV2	1659
15.284.3.27 IID_IAAXControllerV3	1659
15.284.3.28 AAXCompID_PageTableController	1659
15.284.3.29 IID_IAAXPageTableController	1659
15.284.3.30 IID_IAAXPageTableControllerV2	1660
15.284.3.31 AAXCompID_PrivateDataAccess	1660
15.284.3.32 IID_IAAXPrivateDataAccessV1	1660
15.284.3.33 AAXCompID_ViewContainer	1660
15.284.3.34 IID_IAAXViewContainerV1	1660
15.284.3.35 IID_IAAXViewContainerV2	1660
15.284.3.36 IID_IAAXViewContainerV3	1661
15.284.3.37 AAXCompID_Transport	1661
15.284.3.38 IID_IAAXTransportV1	1661
15.284.3.39 IID_IAAXTransportV2	1661
15.284.3.40 IID_IAAXTransportV3	1661
15.284.3.41 IID_IAAXTransportV4	1661
15.284.3.42 AAXCompID_TransportControl	1662

15.284.3.43 IID_IAAXTransportControlV1	1662
15.284.3.44 AAXCompID_PageTable	1662
15.284.3.45 IID_IAAXPageTableV1	1662
15.284.3.46 IID_IAAXPageTableV2	1662
15.284.3.47 AAX_CompID_DescriptionHost	1662
15.284.3.48 IID_IAAXDescriptionHostV1	1663
15.284.3.49 AAX_CompID_FeatureInfo	1663
15.284.3.50 IID_IAAXFeatureInfoV1	1663
15.284.3.51 AAXCompID_Task	1663
15.284.3.52 IID_IAAXTaskV1	1663
15.284.3.53 AAXCompID_SessionDocument	1663
15.284.3.54 IID_IAAXSessionDocumentV1	1664
15.284.3.55 AAXCompID_EffectParameters	1664
15.284.3.56 IID_IAAXEffectParametersV1	1664
15.284.3.57 IID_IAAXEffectParametersV2	1664
15.284.3.58 IID_IAAXEffectParametersV3	1664
15.284.3.59 IID_IAAXEffectParametersV4	1664
15.284.3.60 AAXCompID_HostProcessor	1665
15.284.3.61 IID_IAAXHostProcessorV1	1665
15.284.3.62 IID_IAAXHostProcessorV2	1665
15.284.3.63 AAXCompID_EffectGUI	1665
15.284.3.64 IID_IAAXEffectGUIV1	1665
15.284.3.65 AAXCompID_EffectDirectData	1665
15.284.3.66 IID_IAAXEffectDirectDataV1	1666
15.284.3.67 IID_IAAXEffectDirectDataV2	1666
15.284.3.68 AAXCompID_TaskAgent	1666
15.284.3.69 IID_IAAXTaskAgentV1	1666
15.284.3.70 AAXCompID_SessionDocumentClient	1666
15.284.3.71 IID_IAAXSessionDocumentClientV1	1666
15.284.3.72 AAXCompID_DataBuffer	1667
15.284.3.73 IID_IAAXDataBufferV1	1667
15.284.3.74 AAXATTR_ClientFeature_StemFormat	1667
15.284.3.75 AAXATTR_ClientFeature_AuxOutputStem	1667
15.284.3.76 AAXATTR_ClientFeature_SideChainInput	1667
15.284.3.77 AAXATTR_ClientFeature_MIDI	1668
15.284.3.78 AAXATTR_Client_Level	1668
15.284.3.79 AAX_DocumentDataType_TempoMap	1668
15.285 AAX_UIDs.h	1668
15.286 AAX_UtilsNative.h File Reference	1671
15.286.1 Description	1671
15.286.2 Macro Definition Documentation	1672
15.286.2.1 _AAX_UTILSNATIVE_H_	1672

15.287 AAX_UtilsNative.h	1672
15.288 AAX_VAutomationDelegate.h File Reference	1673
15.288.1 Description	1673
15.289 AAX_VAutomationDelegate.h	1673
15.290 AAX_VCollection.h File Reference	1674
15.290.1 Description	1674
15.291 AAX_VCollection.h	1674
15.292 AAX_VComponentDescriptor.h File Reference	1675
15.292.1 Description	1675
15.293 AAX_VComponentDescriptor.h	1676
15.294 AAX_VController.h File Reference	1677
15.294.1 Description	1677
15.295 AAX_VController.h	1677
15.296 AAX_VDataBufferWrapper.h File Reference	1679
15.296.1 Macro Definition Documentation	1679
15.296.1.1 AAX_VDATABUFFERWRAPPER_H	1679
15.297 AAX_VDataBufferWrapper.h	1680
15.298 AAX_VDescriptionHost.h File Reference	1680
15.299 AAX_VDescriptionHost.h	1680
15.300 AAX_VEffectDescriptor.h File Reference	1681
15.300.1 Description	1681
15.301 AAX_VEffectDescriptor.h	1682
15.302 AAX_Version.h File Reference	1682
15.302.1 Description	1682
15.302.2 Macro Definition Documentation	1683
15.302.2.1 _AAX_VERSION_H_	1683
15.302.2.2 AAX_SDK_VERSION	1683
15.302.2.3 AAX_SDK_CURRENT_REVISION	1684
15.302.2.4 AAX_SDK_1p0p1_REVISION	1684
15.302.2.5 AAX_SDK_1p0p2_REVISION	1684
15.302.2.6 AAX_SDK_1p0p3_REVISION	1684
15.302.2.7 AAX_SDK_1p0p4_REVISION	1684
15.302.2.8 AAX_SDK_1p0p5_REVISION	1684
15.302.2.9 AAX_SDK_1p0p6_REVISION	1684
15.302.2.10 AAX_SDK_1p5p0_REVISION	1685
15.302.2.11 AAX_SDK_2p0b1_REVISION	1685
15.302.2.12 AAX_SDK_2p0p0_REVISION	1685
15.302.2.13 AAX_SDK_2p0p1_REVISION	1685
15.302.2.14 AAX_SDK_2p1p0_REVISION	1685
15.302.2.15 AAX_SDK_2p1p1_REVISION	1685
15.302.2.16 AAX_SDK_2p2p0_REVISION	1685
15.302.2.17 AAX_SDK_2p2p1_REVISION	1685

15.302.2.18 AAX_SDK_2p2p2_REVISION	1686
15.302.2.19 AAX_SDK_2p3p0_REVISION	1686
15.302.2.20 AAX_SDK_2p3p1_REVISION	1686
15.302.2.21 AAX_SDK_2p3p2_REVISION	1686
15.302.2.22 AAX_SDK_2p4p0_REVISION	1686
15.302.2.23 AAX_SDK_2p4p1_REVISION	1686
15.302.2.24 AAX_SDK_2p5p0_REVISION	1686
15.302.2.25 AAX_SDK_2p6p0_REVISION	1686
15.302.2.26 AAX_SDK_2p6p1_REVISION	1687
15.303 AAX_Version.h	1687
15.304 AAX_VFeatureInfo.h File Reference	1687
15.305 AAX_VFeatureInfo.h	1688
15.306 AAX_VHostProcessorDelegate.h File Reference	1688
15.306.1 Description	1688
15.307 AAX_VHostProcessorDelegate.h	1689
15.308 AAX_VHostServices.h File Reference	1689
15.308.1 Description	1689
15.309 AAX_VHostServices.h	1690
15.310 AAX_VPageTable.h File Reference	1690
15.311 AAX_VPageTable.h	1691
15.312 AAX_VPrivateDataAccess.h File Reference	1692
15.312.1 Description	1692
15.313 AAX_VPrivateDataAccess.h	1692
15.314 AAX_VPropertyMap.h File Reference	1693
15.314.1 Description	1693
15.315 AAX_VPropertyMap.h	1693
15.316 AAX_VSessionDocument.h File Reference	1694
15.316.1 Macro Definition Documentation	1694
15.316.1.1 AAX_VSessionDocument_H	1694
15.317 AAX_VSessionDocument.h	1695
15.318 AAX_VTask.h File Reference	1695
15.318.1 Macro Definition Documentation	1696
15.318.1.1 AAX_VTask_H	1696
15.319 AAX_VTask.h	1696
15.320 AAX_VTransport.h File Reference	1696
15.320.1 Description	1697
15.321 AAX_VTransport.h	1697
15.322 AAX_VViewContainer.h File Reference	1698
15.322.1 Description	1698
15.323 AAX_VViewContainer.h	1698
15.324 AAX_Alignment.h File Reference	1699
15.324.1 Description	1699

15.325 AAX_Alignment.h	1700
15.326 AAX_Constants.h File Reference	1700
15.326.1 Description	1700
15.326.2 Macro Definition Documentation	1701
15.326.2.1 AAX_CONSTANTS_H	1701
15.327 AAX_Constants.h	1702
15.328 AAX_Denormal.h File Reference	1703
15.328.1 Description	1703
15.328.2 Macro Definition Documentation	1703
15.328.2.1 AAX_DENORMAL_H	1704
15.328.2.2 AAX_SCOPE_COMPUTE_DENORMALS	1704
15.328.2.3 AAX_SCOPE_DENORMALS_AS_ZERO	1704
15.329 AAX_Denormal.h	1704
15.330 AAX_FastInterpolatedTableLookup.h File Reference	1707
15.330.1 Description	1707
15.330.2 Macro Definition Documentation	1707
15.330.2.1 AAX_FASTINTERPOLATEDTABLELOOKUP_H	1708
15.331 AAX_FastInterpolatedTableLookup.h	1708
15.332 AAX_FastInterpolatedTableLookup.hpp File Reference	1709
15.333 AAX_FastInterpolatedTableLookup.hpp	1709
15.334 AAX_FastPow.h File Reference	1710
15.334.1 Description	1710
15.334.2 Macro Definition Documentation	1711
15.334.2.1 _AAX_FASTPOW_H_	1711
15.335 AAX_FastPow.h	1711
15.336 AAX_Map.h File Reference	1713
15.336.1 Description	1713
15.336.2 Macro Definition Documentation	1713
15.336.2.1 AAX_MAP_H	1713
15.337 AAX_Map.h	1714
15.338 AAX_MiscUtils.h File Reference	1714
15.338.1 Description	1714
15.338.2 Macro Definition Documentation	1715
15.338.2.1 AAX_MISCUTILS_H	1716
15.338.2.2 AAX_ALIGNMENT_HINT	1716
15.338.2.3 AAX_WORD_ALIGNED_HINT	1716
15.338.2.4 AAX_DWORD_ALIGNED_HINT	1716
15.338.2.5 AAX_LO	1716
15.338.2.6 AAX_HI	1716
15.338.2.7 AAX_INT_LO	1717
15.338.2.8 AAX_INT_HI	1717
15.339 AAX_MiscUtils.h	1717

15.340 AAX_PlatformOptimizationConstants.h File Reference	1720
15.340.1 Description	1720
15.340.2 Macro Definition Documentation	1720
15.340.2.1 AAX_PLATFORMOPTIMIZATIONCONSTANTS_H	1720
15.341 AAX_PlatformOptimizationConstants.h	1720
15.342 AAX_Quantize.h File Reference	1721
15.342.1 Description	1721
15.342.2 Macro Definition Documentation	1721
15.342.2.1 AAX_QUANTIZE_H	1721
15.343 AAX_Quantize.h	1722
15.344 AAX_RandomGen.h File Reference	1723
15.344.1 Description	1723
15.344.2 Macro Definition Documentation	1724
15.344.2.1 AAX_RANDOMGEN_H	1724
15.345 AAX_RandomGen.h	1724
15.346 AAX_SampleRateUtils.h File Reference	1725
15.346.1 Description	1725
15.346.2 Enumeration Type Documentation	1726
15.346.2.1 ESRUtils	1726
15.346.3 Function Documentation	1726
15.346.3.1 CoarseSampleRate()	1727
15.346.3.2 CoarseSampleRateFactor()	1727
15.346.3.3 CoarseSampleRateIndex()	1728
15.347 AAX_SampleRateUtils.h	1728

Index**1731**

Chapter 1

Main Page

[AAX SDK Manual](#)

1.1 Welcome to AAX

Select the "Manual" tab to see a full list of documentation pages, or choose from the topics below.

Note

Looking for something? The search function only includes indexing of code symbols and page titles. To search for specific text strings in the AAX SDK manual it is best to use a text search tool such as grep or FINDSTR on the AAX SDK directory or search for the desired text within the PDF version of the AAX SDK documentation.

1.1.1 The Basics

New to AAX? Read through the documentation pages listed below to get started!

- See [Getting Started with AAX](#) for a general overview of AAX and a walk-through of the DemoGain example plug-in
- Read through the first few sections of the [Pro Tools Guide](#) if you are new to Pro Tools
- Read the [Digital signature](#) section of the [Pro Tools Guide](#) to review the digital signing requirements for compatibility with Pro Tools
- Review the [sample plug-ins](#) for examples of both basic and advanced AAX features
- See [Core AAX Interface](#) to find out more about the basic structure of AAX plug-ins
- See [Real-time algorithm callback](#) to learn more about audio processing in AAX
- See [Data model interface](#) to learn about adding parameters and controls to your plug-in,
- See [Description callback](#) for more information about plug-in configuration and initial set-up
- See the [HDX DSP Guide](#) to find out how to add AAX DSP support to your plug-in for Avid's HDX and Pro Tools | Carbon products
- Ready to ship your new plug-in? See [Distributing Your AAX Plug-In](#) for information about finalizing and distributing your AAX products
- Check out the [Troubleshooting](#) page if you're having problems, or post a question to the [developer forums](#) and an Avid engineer will be happy to assist you.

1.1.2 More Topics

Have a more specific question? Review the pages below or view the full list of documentation pages under the "Manual" tab above.

- See [AAX_IController](#) to see the interface that an AAX plug-in's host-based modules use to interact with the host
- See [AAX communication protocols](#) to find out more about how different modules in an AAX plug-in communicate with one another
- See [Offline processing interface](#) for information about creating advanced non-real-time AAX plug-ins
- See [Taper delegates](#) and [Display delegates](#) for more information about implementing custom parameter and control behavior
- Look in the /TI/SignalProcessing folder for signal processing utility classes and functions available for optimizing on Native and DSP

1.1.3 Test Tools & Utilities

- The [DSH Guide](#) has information about the tool for testing basic functionality of your plug-in
- See [DTT Guide](#) to learn how to automate different test scenarios for DSH

1.1.4 Supplemental Information

- [Example Plug-Ins](#)
- [Change Log](#)
- [Host Support](#)
- [Known Issues](#)

1.2 SDK Folder Hierarchy

Documentation

SDK documentation

ExamplePlugins

Example plug-in projects [More information](#)

Extensions

Demonstrations of how to extend the AAX SDK, for example by incorporating third-party GUI frameworks into AAX plug-ins. [More information](#)

Interfaces

Interface headers and other resources required for use of the AAX SDK library

Libs

Source code for the AAX SDK library, a collection of default implementations and utility classes for use in all AAX plug-ins

TI

Various resources for use with TI's Code Composer Studio IDE and Avid's TI testing toolset

Utilities

Common SDK utilities and resources

1.3 Contacting Avid

Your personal [avid user account](#) is your hub for AAX Toolkit services and developer support.

Log in at [avid.com](#) for access to the full range of tools and services provided to AAX developers, including the AAX [developer forum](#). If you have any questions on the AAX SDK documentation or require support with AAX development, we encourage you to post them to the forum as your first line of inquiry.

If you have time-sensitive or critical support inquiries, contact the AAX development team directly at devservices@avid.com. Any AAX questions sent to this alias will be promptly addressed by the most appropriate contact here at Avid.

If you require NFR (Not For Resale) licenses to Avid software for AAX development please send an e-mail to devauth@avid.com with "License Request" in the subject.

If you require access to the digital signing toolkit from PACE Anti-Piracy, Inc. for compatibility with Pro Tools then please follow the instructions [here](#).

The following chart describes these and other ways of connecting with Avid to take advantage of the services provided to AAX developers:

1.4 Licensing

Unless you have entered into a commercial agreement with Avid, you are using this SDK under an evaluation agreement. To review this agreement, see the [AAX Toolkit downloads](#) section under your [my.avid.com](#) account.

As an Avid Developer, you are invited to offer your products on [Avid Marketplace](#) and via [Avid Link](#). If you wish to sell them independently or through other commercial outlets, an authorized representative from your organization is required to sign our Commercial License, which you can read and click through [here](#).

Chapter 2

Todo List

Member [AAX_CAudioInPort](#)

Not used directly by AAX plug-ins

Member [AAX_CAudioOutPort](#)

Not used directly by AAX plug-ins

Class [AAX_CChunkDataParser](#)

Update this documentation for [AAX](#)

Member [AAX_CComponentID](#)

Not used by AAX plug-ins

Member [AAX_CCount](#)

Not used by AAX plug-ins

Member [AAX_CEffectDirectData::Controller](#) (void)

Change to GetController to match other AAX_CEffect modules

Member [AAX_CEffectDirectData::EffectParameters](#) (void)

Change to GetController to match other AAX_CEffect modules

Member [AAX_CEffectGUI::UpdateAllParameters](#) (void)

Rename to `UpdateAllParameterViews()` or another name that does not lead to confusion regarding what exactly this method should be doing.

Member [AAX_CIndex](#)

Not used by AAX plug-ins (except as [AAX_CFieldIndex](#))

Member [AAX_CMeterID](#)

Not used by AAX plug-ins

Member [AAX_CMeterPort](#)

Not used directly by AAX plug-ins

Class [AAX_CParameterManager](#)

Should the Parameter Manager return error codes?

Member [AAX_CParameterManager::AddParameter](#) (AAX_IParameter *param)

Should this method return success/failure code?

Member [AAX_CParameterManager::RemoveAllParameters](#) ()

Should this method return success/failure code?

Member [AAX_CParameterManager::RemoveParameter](#) (AAX_IParameter *param)

Should this method return success/failure code?

Member **AAX_CParameterManager::RemoveParameterByID** (AAX_CParamID identifier)

Should this method return success/failure code?

Member **AAX_CRangeTaperDelegate< T, RealPrecision >::AAX_CRangeTaperDelegate** (T *range, double *rangesSteps, long numRanges, bool useSmartRounding=true)

Document useSmartRounding parameter

Member **AAX_CRangeTaperDelegate< T, RealPrecision >::SmartRound** (double value) const

Document

Member **AAX_CSelector**

Clean up usage; currently used for a variety of ID-related values

Member **AAX_EParameterOrientationBits**

FLAGGED FOR REVISION

Member **AAX_EParameterType**

FLAGGED FOR REMOVAL

Member **AAX_IACFEffEffectGUI::SetControlHighlightInfo** (AAX_CParamID iParameterID, AAX_CBoolean iIsHighlighted, AAX_EHighlightColor iColor)=0

Document this method

Class **AAX_IACFEffEffectParameters**

Add documentation for expected error state return values

Member **AAX_IACFEffEffectParameters::GetParameterOrientation** (AAX_CParamID iParameterID, AAX_EParameterOrientation *oParameterOrientation) const =0

update this documentation

Member **AAX_IACFEffEffectParameters::GetParameterType** (AAX_CParamID iParameterID, AAX_EParameterType *oParameterType) const =0

The concept of parameter type needs more documentation

Member **AAX_IACFEffEffectParameters::SetParameterDefaultNormalizedValue** (AAX_CParamID iParameterID, double iValue)=0

THIS IS NOT CALLED FROM HOST. USEFUL FOR INTERNAL USE ONLY?

Member **AAX_IACFEffEffectParameters::SetParameterNormalizedRelative** (AAX_CParamID iParameterID, double iValue)=0

REMOVE THIS METHOD (?)

NOT CURRENTLY CALLED FROM THE HOST. USEFUL FOR INTERNAL USE ONLY?

Member **AAX_IACFEffEffectParameters::Uninitialize** ()=0

Docs: When exactly is **AAX_IACFEffEffectParameters::Uninitialize()** called, and under what conditions?

Member **AAX_IACFEffEffectParameters::UpdateParameterNormalizedValue** (AAX_CParamID iParameterID, double iValue, AAX_EUpdateSource iSource)=0

FLAGGED FOR CONSIDERATION OF REVISION

Class **AAX_IACFEffEffectParameters_V2**

Add documentation for expected error state return values

Class **AAX_IACFEffEffectParameters_V3**

Add documentation for expected error state return values

Class **AAX_IACFEffEffectParameters_V4**

Add documentation for expected error state return values

Member **AAX_IComponentDescriptor::AddDmaInstance** (AAX_CFieldIndex inFieldIndex, AAX_IDma::EMode inDmaMode)=0

Update the DMA system management such that operation priority can be set arbitrarily

Member **AAX_IComponentDescriptor::AddProcessProc** (AAX_IPropertyMap *inProperties, AAX_CSelector *outProcIDs=NULL, int32_t inProcIDsSize=0)=0

document this parameter Returned array will be NULL-terminated

Member [AAX_IComponentDescriptor::AddProcessProc_Native](#) (AAX_CProcessProc inProcessProc, [AAX_IPropertyMap](#) *inProperties=NULL, AAX_CInstanceInitProc inInstanceInitProc=NULL, AAX_CBackgroundProc inBackgroundProc=NULL, AAX_CSelector *outProcID=NULL)=0

document this parameter

Member [AAX_IComponentDescriptor::AddProcessProc_TI](#) (const char inDLLFileNameUTF8[], const char inProcessProcSymbol[], [AAX_IPropertyMap](#) *inProperties, const char inInstanceInitProcSymbol[]=NULL, const char inBackgroundProcSymbol[]=NULL, AAX_CSelector *outProcID=NULL)=0

document this parameter

Member [AAX_IController::GetCycleCount](#) (AAX_EProperty inWhichCycleCount, AAX_CPropertyValue *outNumCycles) const =0

PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

Member [AAX_IController::SetCycleCount](#) (AAX_EProperty *inWhichCycleCounts, AAX_CPropertyValue *iValues, int32_t numValues)=0

PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

Member [AAX_IDma::IsTransferComplete](#) ()=0

Clarify return value meaning – ambiguity in documentation

Member [AAX_IParameter::GetType](#) () const =0

Document use cases for control type

Member [AAX_IParameter::SetTaperDelegate](#) ([AAX_ITaperDelegateBase](#) &inTaperDelegate, bool inPreserveValue)=0

Document this parameter

Module [additionalFeatures_Sidechain](#)

Is properties->AddProperty (AAX_eProperty_SupportsSideChainInput, true) even necessary?!?! I believe I saw a p.i. that does not declare this...

Module [advancedTopics_parameterUpdates_sequences](#)

Update this section with information about default chunk setting, which is a separate step following the procedure described below.

Member [DBToGain](#) (double dB)

This should be incorporated into parameters' tapers and not called separately

Member [GainToDB](#) (double aGain)

This should be incorporated into parameters' tapers and not called separately

Chapter 3

Host Compatibility Notes

Member [AAX_CMidiPacket::mIsImmediate](#)

This value is not currently set. Use `mTimestamp == 0` to detect immediate packets

Member [AAX_CParameter< T >::AAX_CParameter](#) ([AAX_CParamID](#) identifier, [const AAX_IString &name](#), [T defaultValue](#), [const AAX_ITaperDelegate< T > &taperDelegate](#), [const AAX_IDisplayDelegate< T > &displayDelegate](#), [bool automatable=false](#))

As of Pro Tools 10.2, DAE will check for a matching parameter NAME and not an ID when reading in automation data from a session saved with an AAX plug-ins RTAS/TDM counter part.

As of Pro Tools 11.1, AAE will first try to match ID. If that fails, AAE will fall back to matching by Name.

Module [AAX_DigiTrace_Guide](#)

This feature is available in Pro Tools 12.6 and higher

Member [AAX_eConstraintLocationMask_DLLChipAffinity](#)

This constraint is supported in Pro Tools 10.2 and higher

Member [AAX_eCurveType_Dynamics](#)

Pro Tools requests this curve type for [Dynamics](#) plug-ins only

Member [AAX_eCurveType_EQ](#)

Pro Tools requests this curve type for [EQ](#) plug-ins only

Member [AAX_eCurveType_Reduction](#)

Pro Tools requests this curve type for [Dynamics](#) plug-ins only

Member [AAX_eDataInPortType_Incremental](#)

Supported in Pro Tools 12.5 and higher; when [AAX_eDataInPortType_Incremental](#) is not supported the port will be treated as [AAX_eDataInPortType_Unbuffered](#)

Member [AAX_EHostModeBits](#)

Supported in Venue 5.6 and higher

Member [AAX_eNotificationEvent_ASPreviewState](#)

Supported in Pro Tools 11 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_ASProcessingState](#)

Supported in Pro Tools 11 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_DelayCompensationState](#)

Supported in Pro Tools 12.6 and higher

Member [AAX_eNotificationEvent_EnteringOfflineMode](#)

Supported in Pro Tools 11 and higher

Member [AAX_eNotificationEvent_ExitingOfflineMode](#)

Supported in Pro Tools 11 and higher

Member [AAX_eNotificationEvent_HostModeChanged](#)

Supported in Venue 5.6 and higher

Member [AAX_eNotificationEvent_LogState](#)

Pro Tools currently only sends this notification to the Direct Data object in the plug-in

Member [AAX_eNotificationEvent_MaxViewSizeChanged](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_ParameterNameChanged](#)

Supported in Pro Tools 2023.3 and higher

Member [AAX_eNotificationEvent_PresetOpened](#)

Supported in Pro Tools 11 and higher

Member [AAX_eNotificationEvent_PriorSettingsInvalid](#)

Supported in Venue 5.6 and higher

Member [AAX_eNotificationEvent_SessionBeingOpened](#)

Supported in Pro Tools 11 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_SessionPathChanged](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_SideChainBeingConnected](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_SideChainBeingDisconnected](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_SignalLatencyChanged](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_TrackNameChanged](#)

Supported in Pro Tools 11.2 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_TransportStateChanged](#)

Supported in Pro Tools 2021.10 and higher

Member [AAX_ePlugInStrings_Progress](#)

Not currently supported by Pro Tools

Member [AAX_eProcessingState_BeginPassGroup](#)

AudioSuite pass group notifications are supported starting in Pro Tools 12.0

Member [AAX_eProcessingState_EndPassGroup](#)

AudioSuite pass group notifications are supported starting in Pro Tools 12.0

Member [AAX_eProperty_Constraint_NeverUnload](#)

AAX_eProperty_Constraint_NeverUnload is not currently implemented in DAE or AAE

Member [AAX_eProperty_DestinationTrack](#)

This property is not supported on Media Composer

Member [AAX_eProperty_DisableAudiosuiteReverse](#)

AAX_eProperty_DisableAudiosuiteReverse is not currently implemented

Member [AAX_eProperty_LatencyContribution](#)

Maximum delay compensation limits will vary from host to host. If your plug-in exceeds the delay compensation sample limit for a given AAX host then you should note this limitation in your user documentation. Example limits:

- Pro Tools 9 and higher: 16,383 samples at 44.1/48 kHz, 32,767 samples at 88.2/96 kHz, or 65,534 samples at 176.4/192 kHz
- Media Composer 8.1 and higher: 16,383 samples at 44.1/48 kHz, 32,767 samples at 88.2/96 kHz

Member [AAX_eProperty_OptionalAnalysis](#)

In Media Composer, optional analysis will also be performed automatically before each channel is rendered. See [MCDEV-2904](#)

Member [AAX_eProperty_SideChainStemFormat](#)

[AAX_eProperty_SideChainStemFormat](#) is not currently implemented in DAE or AAE

Currently Pro Tools supports only [AAX_eStemFormat_Mono](#) side chain inputs

Member [AAX_eProperty_UsesClientGUI](#)

Currently supported by Pro Tools only

Member [AAX_IACFEffEffectParameters::CompareActiveChunk](#) (const [AAX_SPlugInChunk](#) *iChunkP, [AAX_CBoolean](#) *olsEqual) const =0

In Pro Tools, this method will only be called if a prior call to [GetNumberOfChanges\(\)](#) has indicated that the plug-in's state has changed. If the plug-in's current settings are different from the settings in [aChunkP](#) then the plug-in's Compare Light will be illuminated in the plug-in header, allowing users to toggle between the plug-in's custom state and its saved state.

Member [AAX_IACFEffEffectParameters::GetCurveData](#) ([AAX_CTypeID](#) iCurveType, const float *iValues, [uint32_t](#) iNumValues, float *oValues) const =0

Versions of S6 software which support the [GetCurveDataDisplayRange\(\)](#) method will not display a plug-in's curve data unless both [GetCurveData\(\)](#) and [GetCurveDataDisplayRange\(\)](#) are supported by the plug-in.

Member [AAX_IACFEffEffectParameters::GetParameterNameOfLength](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName, [int32_t](#) iNameLength) const =0

In most cases, the AAX host will call [GetParameterName\(\)](#) or [GetParameterNameOfLength\(\)](#) to retrieve parameter names for display. However, when Pro Tools is retrieving a plug-in name for display on a control surface the XML data stored in the plug-in's page tables will be used in preference to values retrieved from these methods.

Member [AAX_IComponentDescriptor::AddAuxOutputStem](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inStemFormat, const char inNameUTF8[]) =0

Pro Tools supports only mono and stereo auxiliary output stem formats

There is a hard limit to the number of outputs that Pro Tools supports for a single plug-in instance. This limit is currently set at 256 channels, which includes all of the plug-in's output channels in addition to the sum total of all of its aux output stem channels.

Member [AAX_IComponentDescriptor::AddClock](#) ([AAX_CFieldIndex](#) inFieldIndex) =0

As of Pro Tools 11.1, this field may be used in both Native and DSP plug-ins. The DSP clock data is a 16-bit cycling counter. This field was only available for Native plug-ins in previous Pro Tools versions.

Member [AAX_IComponentDescriptor::AddMIDINode](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_EMIDINodeType](#) inNodeType, const char inNodeName[], [uint32_t](#) channelMask) =0

Due to current restrictions MIDI data won't be delivered to DSP algorithms, only to AAX Native.

Member [AAX_IController::GetHostName](#) ([AAX_IString](#) *outHostNameString) const =0

Pro Tools versions from Pro Tools 11.0 to Pro Tools 12.3.1 will return a generic version string to this call. This issue is resolved beginning in Pro Tools 12.4.

Member [AAX_IMIDINode::PostMIDIpacket](#) ([AAX_CMidiPacket](#) *packet) =0

Pro Tools supports the following MIDI events from plug-ins:

- NoteOn
- NoteOff

- Pitch bend
- Polyphonic key pressure
- Bank select (controller #0)
- Program change (no bank)
- Channel pressure

Member [AAX_ITransport::GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t *SampleLocation) const =0

There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Member [AAX_ITransport::GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0

This does not indicate anything about the status of the "Loop Record" option. Even when the host is configured to loop playback, looping may not occur if certain conditions are not met (i.e. the length of the selection is too short)

Member [AAX_ITransport::GetCurrentTickPosition](#) (int64_t *TickPosition) const =0

The tick resolution here is different than that of the tick displayed in Pro Tools. "Display ticks" (as they are called) are 1/960 of a quarter note.

Member [AAX_ITransport::GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0

There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Member [AAX_IViewContainer::GetModifiers](#) (uint32_t *outModifiers)=0

Although this method allows plug-ins to acquire the current state of the Windows key (normally blocked by Pro Tools), plug-ins should not use key combinations that require this key.

Module [AAX_Media_Composer_Guide](#)

Some early versions of Media Composer 8 do not search the system plug-ins directory recursively. If your plug-ins are installed into a sub-directory beneath this main directory then they will not be loaded by the affected versions of Media Composer.

Module [AAX_Page_Table_Guide](#)

Pro Tools versions prior to Pro Tools 11.1 use plug-ins' ProControl and ICON page tables (Dynamics, EQ, Channel Strip, Custom Fader, etc.) to map plug-in parameters to EUCON-enabled surfaces, so be sure that your plug-ins also implement these page tables correctly so that users with earlier versions of Pro Tools can have the best possible experience when using your plug-ins.

Module [AAX_Pro_Tools_Guide](#)

Pro Tools requires PACE Eden digital signatures for AAX plug-ins.

Supported in Pro Tools 2019.XX and higher. Also supported (and enabled by default) in Pro Tools developer builds beginning with Pro Tools 2019.6.

Module [AAX_TI_Guide](#)

32 and 64-sample quantum is available in Pro Tools 10.2 and higher

Beginning in Pro Tools 11, AAX DSP algorithms also support optional temporary data spaces that can be described in the Describe module and are shared among all instances on a DSP. This is an alternative to declaring large data blocks on the stack for better memory management and to prevent stack overflows. Please refer to [AAX_IComponentDescriptor::AddTemporaryData\(\)](#) for usage instructions.

Module [AdditionalFeatures_CurveDisplays](#)

For S6 control surface displays, see [PT-226228](#) and [PT-226227](#) in the [Known Issues](#) page for more information about the requirements listed in this section.

Module [advancedTopics_relatedTypes](#)

Pro Tools versions prior to Pro Tools 12.3 do not allow explicit type conversion between types with different product ID values. Beginning in Pro Tools 12.3 both the product ID and the plug-in ID may differ between explicitly related types.

Module [AuxInterface_TaskAgent](#)

This interface is not yet used in any AAX hosts

Module [CommonInterface_Algorithm](#)

As of Pro Tools 10.2.1 an algorithm's initialization callback routine will have up to 5 seconds to execute.

Module [CommonInterface_FormatSpecification](#)

*_ACFGetSDKVersion is required for 64-bit AAX plug-ins only

Module [ExamplePlugins](#)

The DemoDelay_DynamicLatencyComp example is compatible with Pro Tools 11.1 and higher.

Chapter 4

Legacy Porting Notes

Class [AAX_CEffectGUI](#)

The default implementations in this class are mostly derived from their equivalent implementations in CProcess and CEffectProcess. For additional CProcess-derived implementations, see [AAX_CEffectParameters](#).

Class [AAX_CEffectParameters](#)

The default implementations in this class are mostly derived from their equivalent implementations in CProcess and CEffectProcess. For additional CProcess-derived implementations, see [AAX_CEffectGUI](#).

Member [AAX_CHostProcessor::AnalyzeAudio](#) (const float *const inAudiolns[], int32_t inAudiolnCount, int32_t *ioWindowSize) [AAX_OVERRIDE](#)

Ported from AudioSuite's AnalyzeAudio(bool isMasterBypassed) method

Member [AAX_CHostProcessor::InitOutputBounds](#) (int64_t iSrcStart, int64_t iSrcEnd, int64_t *oDstStart, int64_t *oDstEnd) [AAX_OVERRIDE](#)

DAE no longer makes use of the mStartBound and mEndBounds member variables that existed in the legacy RTAS/TDM SDK. Use oDstStart and oDstEnd instead (preferably by overriding [TranslateOutputBounds\(\)](#).)

Member [AAX_CHostProcessor::RenderAudio](#) (const float *const inAudiolns[], int32_t inAudiolnCount, float *const iAudioOuts[], int32_t iAudioOutCount, int32_t *ioWindowSize) [AAX_OVERRIDE](#)

This method is a replacement for the AudioSuite ProcessAudio method

Class [AAX_CMidiPacket](#)

Corresponds to DirectMidiPacket in the legacy SDK

Class [AAX_CMidiStream](#)

Corresponds to DirectMidiNode in the legacy SDK

Member [AAX_eMIDINodeType_Global](#)

Corresponds to RTAS Shared Buffer global nodes in the legacy SDK

Member [AAX_eMIDINodeType_LocalInput](#)

Corresponds to RTAS Buffered MIDI input nodes in the legacy SDK

Member [AAX_eMIDINodeType_LocalOutput](#)

Corresponds to RTAS Buffered MIDI output nodes in the legacy SDK

Member [AAX_eNotificationEvent_ASPreviewState](#)

Replacement for SetPreviewState()

Member [AAX_ePageTable_EQ_Band_Type](#)

converted from eDigi_PageTable_EQ_Band_Type in the legacy SDK

Member [AAX_ePageTable_EQ_InCircuitPolarity](#)

converted from eDigi_PageTable_EQ_InCircuitPolarity in the legacy SDK

Member [AAX_ePageTable_UseAlternateControl](#)

converted from `eDigi_PageTable_UseAlternateControl` in the legacy SDK

Member [AAX_EParameterType](#)

Values must match unnamed type enum in `FicTDMControl.h`

Member [AAX_eParameterType_Continuous](#)

Matches `kDAE_ContinuousValues`

Member [AAX_eParameterType_Discrete](#)

Matches `kDAE_DiscreteValues`

Member [AAX_EParameterValueInfoSelector](#)

converted from `EControlValueInfo` in the legacy SDK

Member [AAX_ePluginStrings_AllSelectedRegionsAnalysis](#)

Was `pluginStrings_AllSelectedRegionsAnalysis` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_Analysis](#)

Was `pluginStrings_Analysis` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_Bypass](#)

Was `pluginStrings_Bypass` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_ClipName](#)

Was `pluginStrings_RegionName` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_MonoMode](#)

Was `pluginStrings_MonoMode` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_MultiInputMode](#)

Was `pluginStrings_MultiInputMode` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_Process](#)

Was `pluginStrings_Process` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_Progress](#)

Was `pluginStrings_Progress` in the RTAS/TDM SDK

Member [AAX_ePluginStrings_RegionByRegionAnalysis](#)

Was `pluginStrings_RegionByRegionAnalysis` in the RTAS/TDM SDK

Member [AAX_EProperty](#)

These property IDs are somewhat analogous to the `pluginGestalt` system in the legacy SDK, and several [AAX_EProperty](#) values correlate directly with a corresponding legacy plug-in gestalt.

To ensure session interchange compatibility, make sure the 4 character IDs for [AAX_eProperty_ManufacturerID](#), [AAX_eProperty_ProductID](#), [AAX_eProperty_PluginID_Native](#), and [AAX_eProperty_PluginID_AudioSuite](#) are identical to the legacy SDK's counterpart.

Member [AAX_eProperty_AllowPreviewWithoutAnalysis](#)

Was `pluginGestalt_AnalyzeOnTheFly`

Member [AAX_eProperty_CanBypass](#)

Was `pluginGestalt_CanBypass`.

Member [AAX_eProperty_ContinuousOnly](#)

Was `pluginGestalt_ContinuousOnly`

Member [AAX_eProperty_DestinationTrack](#)

Was `pluginGestalt_DestinationTrack`

Member [AAX_eProperty_DisablePreview](#)

Was `pluginGestalt_DisablePreview`

Member [AAX_eProperty_DoesntIncrOutputSample](#)

Was `pluginGestalt_DoesntIncrOutputSample`

Member [AAX_eProperty_ManufacturerID](#)

For legacy plug-in session compatibility, this ID should match the Manufacturer ID used in the corresponding legacy plug-ins.

Member [AAX_eProperty_MultiInputModeOnly](#)

Was pluginGestalt_MultiInputModeOnly

Member [AAX_eProperty_NeedsOutputDithered](#)

Was pluginGestalt_NeedsOutputDithered

Member [AAX_eProperty_OptionalAnalysis](#)

Was pluginGestalt_OptionalAnalysis

Member [AAX_eProperty_PluginID_AudioSuite](#)

For legacy plug-in session compatibility, this ID should match the Type ID used in the corresponding legacy AudioSuite plug-in Types.

Member [AAX_eProperty_PluginID_Native](#)

For legacy plug-in session compatibility, this ID should match the Type ID used in the corresponding legacy RTAS plug-in Types.

Member [AAX_eProperty_PluginID_Ti](#)

For legacy plug-in session compatibility, this ID should match the Type ID used in the corresponding legacy TDM plug-in Types.

Member [AAX_eProperty_ProductID](#)

For legacy plug-in session compatibility, this ID should match the Product ID used in the corresponding legacy plug-in.

Member [AAX_eProperty_RequestsAllTrackData](#)

Was pluginGestalt_RequestsAllTrackData

Member [AAX_eProperty_RequiresAnalysis](#)

Was pluginGestalt_RequiresAnalysis

Member [AAX_eProperty_SupportsSaveRestore](#)

Was pluginGestalt_SupportsSaveRestore

Member [AAX_eProperty_UsesRandomAccess](#)

Was pluginGestalt_UsesRandomAccess

Class [AAX_IACFEffEffectGUI](#)

In the legacy plug-in SDK, these methods were found in CProcess and CEffectProcess. For additional CProcess methods, see [AAX_IEffectParameters](#).

Member [AAX_IACFEffEffectGUI::SetControlHighlightInfo](#) (AAX_CParamID iParameterID, AAX_CBoolean iIsHighlighted, AAX_EHighlightColor iColor)=0

This method was re-named from `SetControlHighliteInfo()`, its name in the legacy plug-in SDK.

Member [AAX_IACFEffEffectParameters::GetChunkSize](#) (AAX_CTypeID iChunkID, uint32_t *oSize) const =0

In [AAX](#), the value provided by `GetChunkSize()` should *NOT* include the size of the chunk header. The value should *ONLY* reflect the size of the chunk's data.

Member [AAX_IACFEffEffectParameters::GetParameterOrientation](#) (AAX_CParamID iParameterID, AAX_EParameterOrientation *oParameterOrientation) const =0

[AAX_IEffectParameters::GetParameterOrientation\(\)](#) corresponds to the `GetControlOrientation()` method in the legacy RTAS/TDM SDK.

Member [AAX_IACFEffEffectParameters::GetParameterStringFromValue](#) (AAX_CParamID iParameterID, double iValue, AAX_IString *oValueString, int32_t iMaxLength) const =0

This method corresponds to `CProcess::MapControlValToString()` in the RTAS/TDM SDK

Member [AAX_IACFEffEffectParameters::GetParameterValueFromString](#) (AAX_CParamID iParameterID, double *oValue, const AAX_IString &iValueString) const =0

This method corresponds to `CProcess::MapControlStringToVal()` in the RTAS/TDM SDK

Class [AAX_IACFHostProcessor](#)

This interface provides offline processing features analogous to the legacy AudioSuite plug-in architecture

Class [AAX_ICollection](#)

The information in [AAX_ICollection](#) is roughly analogous to the information provided by `CProcessGroup` in the legacy plug-in library

Class [AAX_IEffectParameters](#)

In the legacy plug-in SDK, these methods were found in `CProcess` and `CEffectProcess`. For additional `CProcess` methods, see [AAX_IEffectGUI](#).

File [AAX_SliderConversions.h](#)

These utilities may be required in order to maintain settings chunk compatibility with plug-ins that were ported from the legacy RTAS/TDM format.

Class [AAX_SPlugInChunkHeader](#)

To ensure compatibility with TDM/RTAS plug-ins whose implementation requires `fSize` to be equal to the size of the chunk's header plus its data, AAE performs some behind-the-scenes record keeping.

The following actions are only taken for AAX plug-ins, so, e.g., if a chunk is stored by an RTAS or TDM plug-in that reports data+header size in `fSize` and this chunk is then loaded by the AAX version of the plug-in, the header size will be cached as-is from the legacy plug-in and will be subtracted out before the chunk data is passed to the AAX plug-in. If a chunk is stored by an AAX plug-in and is then loaded by a legacy plug-in, the legacy plug-in will receive the cached plug-in header with `fSize` equal to the data+header size.

These are the special actions that AAE takes to ensure backwards-compatibility when handling AAX chunk data:

- When AAE retrieves the size of a chunk from an AAX plug-in using [GetChunkSize\(\)](#), it adds the chunk header size to the amount of memory that it allocates for the chunk
- When AAE retrieves a chunk from an AAX plug-in using [GetChunk\(\)](#), it adds the chunk header size to `fChunkSize` before caching the chunk
- Before calling [SetChunk\(\)](#) or [CompareActiveChunk\(\)](#), AAE subtracts the chunk header size from the cached chunk's header's `fChunkSize` member

Module [AdditionalFeatures_Meters](#)

The gain-reduction meter handling for AAX plug-ins is different from that for RTAS/TDM plug-ins. AAX plug-ins must invert their gain-reduction meter values manually before reporting these values from the audio processing callback. The AAX host will always thin reported meter data using a "max" operation, and will later invert gain-reduction meter values before they are available to the plug-in GUI or to control surfaces.

Chapter 5

Deprecated List

Member [AAX::FastRndDbl2Int32](#) (double iVal)

Member [AAX_CInitPrivateDataProc](#)

Member [AAX_CPacketAllocator](#)

Member [AAX_CTaskAgent::AddTask](#) (std::unique_ptr< AAX_ITask > iTask)
Use [ReceiveTask\(\)](#) instead

Member [AAX_EHostMode](#)
This enum is deprecated and will be removed in a future release.

Member [AAX_eHostMode_Config](#)
Use [AAX_eHostModeBits_None](#)

Member [AAX_eHostMode_Show](#)
Use [AAX_eHostModeBits_Live](#)

Member [AAX_ePlugInStrings_PluginFileName](#)

Member [AAX_ePlugInStrings_Preview](#)

Member [AAX_ePlugInStrings_RegionName](#)

Member [AAX_eProcessingState_Start](#)

Member [AAX_eProcessingState_Stop](#)

Member [AAX_eProperty_AudioBufferLength](#)
Use [AAX_eProperty_DSP_AudioBufferLength](#)

Member [AAX_eProperty_Deprecated_Plugin_List](#)
Use [AAX_eProperty_Deprecated_Native_Plugin_List](#) and [AAX_eProperty_Deprecated_DSP_Plugin_List](#) See [AAX_eProperty_PluginID_RTAS](#) for an example.

Member [AAX_eProperty_PluginID_RTAS](#)
Use [AAX_eProperty_PluginID_Native](#)

Member [AAX_eProperty_StoreXMLPageTablesByType](#)

Use [AAX_eProperty_StoreXMLPageTablesByEffect](#)

Member [AAX_IACFEfffectParameters::UpdateParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0

This is not called from the host. It *may* still be useful for internal calls within the plug-in, though it should only ever be used to update non-automatable parameters. Automatable parameters should always be updated through the [AAX_IParameter](#) interface, which will ensure proper coordination with other automation clients.

Member [AAX_IACFHostServices::Assert](#) (const char *iFile, int32_t iLine, const char *iNote)=0

Legacy version of [AAX_IACFHostServices_V3::HandleAssertFailure\(\)](#) implemented by older hosts

Module [ExamplePlugins](#)

The DemoGain_Delay example is deprecated. See DemoDelay_HostProcessor

Chapter 6

Not Used by AAX Plug-Ins

Member [AAX_eUpdateSource_Delay](#)

Chapter 7

Module Index

7.1 Manual

These pages provide further information about various aspects of AAX.

AAX SDK Manual	43
Getting Started with AAX	47
Core AAX Interface	55
Description callback	56
Real-time algorithm callback	66
Data model interface	72
GUI interface	74
AAX communication protocols	76
AAX Format Specification	78
Additional AAX features	80
Direct data access interface	81
Offline processing interface	83
Hybrid Processing architecture	84
MIDI	87
Plug-in meters	90
Sidechain Inputs	92
Auxiliary Output Stems	93
Direct Memory Access	94
Background processing callback	96
EQ and Dynamics Curve Displays	98
Task agent interface	103
AAX Library features	105
Parameter Manager	105
Taper delegates	108
Display delegates	109
Display delegate decorators	110
Additional Topics	111
Real-time performance	112
Parameter automation	113
Parameter updates	115
Parameter update timing	116
Token protocol	123
Basic parameter update sequences	127
Linked parameters	131

Linked parameter update sequences	136
Plug-in type conversion	140
The Avid Component Framework (ACF)	144
ACF Elements	149
AAX Host Guides	150
Pro Tools Guide	151
Media Composer Guide	170
HDX DSP Guide	178
Page Table Guide	224
DigiTrace Guide	260
DSH Guide	271
DTT Guide	277
VENUE Guide	355
Extensions	281
GUI Extensions	282
Monolithic VIs and Effects	283
Other Extensions	284
Supplemental Information	285
Troubleshooting	286
Distributing Your AAX Plug-In	290
AAX Interfaces	294
Host Support	295
Known Issues	302
Change Log	334
Example Plug-Ins	352

Chapter 8

Namespace Index

8.1 Namespace List

Here is a list of all namespaces with brief descriptions:

AAX	371
AAX::Exception	
AAX exception classes	406
AAX::internal	407
AAX_ChunkDataParserDefs	
Constants used by ChunkDataParser class	407

Chapter 9

Hierarchical Index

9.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

_acfUID	411
AAX_AggregateResult	412
AAX_CAutoreleasePool	429
AAX_CChunkDataParser	439
AAX_CheckedResult	506
AAX_CHostServices	525
AAX_CMidiPacket	538
AAX_CMidiStream	540
AAX_CMutex	558
AAX_Component< aContextType >	565
AAX_CPacket	566
AAX_CPacketDispatcher	568
AAX_CParameterManager	615
AAX_CStringAbbreviations	707
AAX_CTempoBreakpoint	728
AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >	740
AAX_IAutomationDelegate	940
AAX_VAutomationDelegate	1163
AAX_ICollection	946
AAX_VCollection	1168
AAX_IComponentDescriptor	951
AAX_VComponentDescriptor	1175
AAX_IContainer	968
AAX_IPointerQueue< TNumberedParamStateList >	1089
AAX_IPointerQueue< const TParamValPair >	1089
AAX_IPointerQueue< T >	1089
AAX_CAtomicQueue< TNumberedParamStateList, 256 >	424
AAX_CAtomicQueue< const TParamValPair, 16 *kSynchronizedParameterQueueSize >	424
AAX_CAtomicQueue< T, S >	424
AAX_IController	970
AAX_VController	1190
AAX_IDataBufferWrapper	988
AAX_VDataBufferWrapper	1207

AAX_IDescriptionHost	990
AAX_VDescriptionHost	1210
AAX_IDisplayDelegateBase	996
AAX_IDisplayDelegate< T >	991
AAX_CBinaryDisplayDelegate< T >	430
AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >	560
AAX_CStateDisplayDelegate< T >	653
AAX_CStringDisplayDelegate< T >	718
AAX_IDisplayDelegateDecorator< T >	997
AAX_CDecibelDisplayDelegateDecorator< T >	446
AAX_CPercentDisplayDelegateDecorator< T >	630
AAX_CUnitDisplayDelegateDecorator< T >	728
AAX_CUnitPrefixDisplayDelegateDecorator< T >	734
AAX_IDma	1003
AAX_IEffectDescriptor	1014
AAX_VEffectDescriptor	1213
AAX_IFeatureInfo	1033
AAX_VFeatureInfo	1219
AAX_IHostProcessorDelegate	1038
AAX_VHostProcessorDelegate	1222
AAX_IHostServices	1041
AAX_VHostServices	1225
AAX_IMIDIMessageInfoDelegate	1044
AAX_IMIDINode	1048
AAX_IPacketHandler	1050
AAX_CPacketHandler< TWorker >	572
AAX_IPageTable	1051
AAX_VPageTable	1228
AAX_IParameter	1063
AAX_CParameter< T >	575
AAX_CStatelessParameter	658
AAX_IParameterValue	1085
AAX_CParameterValue< T >	621
AAX_IPrivateDataAccess	1092
AAX_VPrivateDataAccess	1241
AAX_IPropertyMap	1094
AAX_VPropertyMap	1243
AAX_ISessionDocument	1100
AAX_VSessionDocument	1250
AAX_IString	1104
AAX_CString	690
AAX_ITaperDelegateBase	1111
AAX_ITaperDelegate< T >	1107
AAX_CBinaryTaperDelegate< T >	435
AAX_CLinearTaperDelegate< T, RealPrecision >	527
AAX_CLogTaperDelegate< T, RealPrecision >	533
AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >	636
AAX_CRangeTaperDelegate< T, RealPrecision >	641
AAX_CStateTaperDelegate< T >	686
AAX_ITask	1113
AAX_VTask	1252
AAX_ITransport	1119
AAX_VTransport	1256

AAX_IViewContainer	1128
AAX_VViewContainer	1267
AAX_Map	1134
AAX_Point	1137
AAX_Rect	1138
AAX_SHybridRenderInfo	1140
AAX_SInstrumentPrivateData	1141
AAX_SInstrumentRenderInfo	1142
AAX_SInstrumentSetupInfo	1145
AAX_SPlugInChunk	1153
AAX_SPlugInChunkHeader	1156
AAX_SPlugInIdentifierTriad	1158
AAX_StLock_Guard	1159
AAX_TransportStateInfo_V1	1161
AAX::Exception::Any	1273
AAX::Exception::ResultError	1284
CACFUnknown	
AAX_IDataBuffer	986
AAX_CArrayDataBuffer< D >	416
AAX_CArrayDataBufferOfType< T, D >	420
AAX_CStringDataBuffer	710
AAX_CStringDataBufferOfType< T >	714
AAX_IEffectDirectData	1020
AAX_CEffectDirectData	452
AAX_IEffectGUI	1024
AAX_CEffectGUI	458
AAX_IEffectParameters	1027
AAX_CEffectParameters	469
AAX_CMonolithicParameters	541
AAX_IHostProcessor	1035
AAX_CHostProcessor	511
AAX_ISessionDocumentClient	1102
AAX_CSessionDocumentClient	647
AAX_ITaskAgent	1116
AAX_CTaskAgent	723
AAX_CChunkDataParser::DataValue	1277
IACFPluginDefinition	
AAX_IACFCollection	746
IACFUnknown	1282
AAX_IACFAutomationDelegate	743
AAX_IACFComponentDescriptor	749
AAX_IACFComponentDescriptor_V2	759
AAX_IACFComponentDescriptor_V3	762
AAX_IACFController	766
AAX_IACFController_V2	775
AAX_IACFController_V3	779
AAX_IACFDataBuffer	783
AAX_IDataBuffer	986
AAX_IACFDescriptionHost	785
AAX_IACFEffectDescriptor	787
AAX_IACFEffectDescriptor_V2	792
AAX_IACFEffectDirectData	794
AAX_IACFEffectDirectData_V2	797
AAX_IEffectDirectData	1020
AAX_IACFEffectGUI	799
AAX_IEffectGUI	1024

AAX_IACFEffEffectParameters	806
AAX_IACFEffEffectParameters_V2	830
AAX_IACFEffEffectParameters_V3	836
AAX_IACFEffEffectParameters_V4	842
AAX_IEffEffectParameters	1027
AAX_IACFFeatureInfo	848
AAX_IACFHostProcessor	850
AAX_IACFHostProcessor_V2	857
AAX_IHostProcessor	1035
AAX_IACFHostProcessorDelegate	859
AAX_IACFHostProcessorDelegate_V2	862
AAX_IACFHostProcessorDelegate_V3	864
AAX_IACFHostServices	866
AAX_IACFHostServices_V2	868
AAX_IACFHostServices_V3	870
AAX_IACFPageTable	872
AAX_IACFPageTable_V2	878
AAX_IACFPageTableController	884
AAX_IACFPageTableController_V2	888
AAX_IACFPrivateDataAccess	891
AAX_IACFPropertyMap	893
AAX_IACFPropertyMap_V2	895
AAX_IACFPropertyMap_V3	898
AAX_IACFSessionDocument	901
AAX_IACFSessionDocumentClient	902
AAX_ISessionDocumentClient	1102
AAX_IACFTask	905
AAX_IACFTaskAgent	909
AAX_ITaskAgent	1116
AAX_IACFTransport	912
AAX_IACFTransport_V2	917
AAX_IACFTransport_V3	921
AAX_IACFTransport_V4	924
AAX_IACFTransportControl	928
AAX_IACFViewContainer	930
AAX_IACFViewContainer_V2	934
AAX_IACFViewContainer_V3	937
IACFDefinition	1279
SArray< T >	1287
AAX_ISessionDocument::TempoMap	1288
AAX_VSessionDocument::VTempoMap	1290

Chapter 10

Class Index

10.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_acfUID	411
AAX_AggregateResult	412
AAX_CArrayDataBuffer< D >	416
AAX_CArrayDataBufferOfType< T, D >	
A convenience class for array data buffers	420
AAX_CAtomicQueue< T, S >	424
AAX_CAutoreleasePool	429
AAX_CBinaryDisplayDelegate< T >	
A binary display format conforming to AAX_IDisplayDelegate	430
AAX_CBinaryTaperDelegate< T >	
A binary taper conforming to AAX_ITaperDelegate	435
AAX_CChunkDataParser	
Parser utility for plugin chunks	439
AAX_CDecibelDisplayDelegateDecorator< T >	
A percent decorator conforming to AAX_IDisplayDelegateDecorator	446
AAX_CEffectDirectData	
Default implementation of the AAX_IEffectDirectData interface	452
AAX_CEffectGUI	
Default implementation of the AAX_IEffectGUI interface	458
AAX_CEffectParameters	
Default implementation of the AAX_IEffectParameters interface	469
AAX_CheckedResult	506
AAX_CHostProcessor	
Concrete implementation of the AAX_IHostProcessor interface for non-real-time processing	511
AAX_CHostServices	
Method access to a singleton implementation of the AAX_IHostServices interface	525
AAX_CLinearTaperDelegate< T, RealPrecision >	
A linear taper conforming to AAX_ITaperDelegate	527
AAX_CLogTaperDelegate< T, RealPrecision >	
A logarithmic taper conforming to AAX_ITaperDelegate	533
AAX_CMidiPacket	
Packet structure for MIDI data	538
AAX_CMidiStream	
MIDI stream data structure used by AAX_IMIDINode	540
AAX_CMonolithicParameters	
Extension of the AAX_CEffectParameters class for monolithic VIs and effects	541

AAX_CMutex	
Mutex with try lock functionality	558
AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >	
A numeric display format conforming to AAX_IDisplayDelegate	560
AAX_Component< aContextType >	
Empty class containing type declarations for the AAX algorithm and associated callbacks	565
AAX_CPacket	
Container for packet-related data	566
AAX_CPacketDispatcher	
Helper class for managing AAX packet posting	568
AAX_CPacketHandler< TWorker >	
Callback container used by AAX_CPacketDispatcher	572
AAX_CParameter< T >	
Generic implementation of an AAX_IParameter	575
AAX_CParameterManager	
A container object for plug-in parameters	615
AAX_CParameterValue< T >	
Concrete implementation of AAX_IParameterValue	621
AAX_CPercentDisplayDelegateDecorator< T >	
A percent decorator conforming to AAX_IDisplayDelegateDecorator	630
AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >	
A piece-wise linear taper conforming to AAX_ITaperDelegate	636
AAX_CRangeTaperDelegate< T, RealPrecision >	
A piecewise-linear taper conforming to AAX_ITaperDelegate	641
AAX_CSessionDocumentClient	
Default implementation of the AAX_ISessionDocumentClient interface	647
AAX_CStateDisplayDelegate< T >	
A generic display format conforming to AAX_IDisplayDelegate	653
AAX_CStatelessParameter	
A stateless parameter implementation	658
AAX_CStateTaperDelegate< T >	
A linear taper conforming to AAX_ITaperDelegate	686
AAX_CString	
A generic AAX string class with similar functionality to <code>std::string</code>	690
AAX_CStringAbbreviations	
Helper class to store a collection of name abbreviations	707
AAX_CStringDataBuffer	710
AAX_CStringDataBufferOfType< T >	
A convenience class for string data buffers	714
AAX_CStringDisplayDelegate< T >	
A string, or list, display format conforming to AAX_IDisplayDelegate	718
AAX_CTaskAgent	
Default implementation of the AAX_ITaskAgent interface	723
AAX_CTempoBreakpoint	728
AAX_CUnitDisplayDelegateDecorator< T >	
A unit type decorator conforming to AAX_IDisplayDelegateDecorator	728
AAX_CUnitPrefixDisplayDelegateDecorator< T >	
A unit prefix decorator conforming to AAX_IDisplayDelegateDecorator	734
AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >	740
AAX_IACFAutomationDelegate	
Versioned interface allowing an AAX plug-in to interact with the host's automation system	743
AAX_IACFCollection	
Versioned interface to represent a plug-in binary's static description	746
AAX_IACFComponentDescriptor	
Versioned description interface for an AAX plug-in algorithm callback	749
AAX_IACFComponentDescriptor_V2	
Versioned description interface for an AAX plug-in algorithm callback	759

AAX_IACFComponentDescriptor_V3	
Versioned description interface for an AAX plug-in algorithm callback	762
AAX_IACFController	
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components	766
AAX_IACFController_V2	
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.	775
AAX_IACFController_V3	
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.	779
AAX_IACFDataBuffer	
Versioned interface for reference counted data buffers	783
AAX_IACFDescriptionHost	785
AAX_IACFEffectDescriptor	
Versioned interface for an AAX_IEffectDescriptor	787
AAX_IACFEffectDescriptor_V2	
Versioned interface for an AAX_IEffectDescriptor	792
AAX_IACFEffectDirectData	
Optional interface for direct access to a plug-in's alg memory	794
AAX_IACFEffectDirectData_V2	797
AAX_IACFEffectGUI	
The interface for a AAX Plug-in's GUI (graphical user interface)	799
AAX_IACFEffectParameters	
The interface for an AAX Plug-in's data model	806
AAX_IACFEffectParameters_V2	
Supplemental interface for an AAX Plug-in's data model	830
AAX_IACFEffectParameters_V3	
Supplemental interface for an AAX Plug-in's data model	836
AAX_IACFEffectParameters_V4	
Supplemental interface for an AAX Plug-in's data model	842
AAX_IACFFeatureInfo	848
AAX_IACFHostProcessor	
Versioned interface for an AAX host processing component	850
AAX_IACFHostProcessor_V2	
Supplemental interface for an AAX host processing component	857
AAX_IACFHostProcessorDelegate	
Versioned interface for host methods specific to offline processing	859
AAX_IACFHostProcessorDelegate_V2	
Versioned interface for host methods specific to offline processing	862
AAX_IACFHostProcessorDelegate_V3	
Versioned interface for host methods specific to offline processing	864
AAX_IACFHostServices	
Versioned interface to diagnostic and debugging services provided by the AAX host	866
AAX_IACFHostServices_V2	
V2 of versioned interface to diagnostic and debugging services provided by the AAX host . . .	868
AAX_IACFHostServices_V3	
V3 of versioned interface to diagnostic and debugging services provided by the AAX host . . .	870
AAX_IACFPageTable	
Versioned interface to the host's representation of a plug-in instance's page table	872
AAX_IACFPageTable_V2	
Versioned interface to the host's representation of a plug-in instance's page table	878
AAX_IACFPageTableController	
Interface for host operations related to the page tables for this plug-in	884
AAX_IACFPageTableController_V2	
Interface for host operations related to the page tables for this plug-in.	888
AAX_IACFPrivateDataAccess	
Interface for the AAX host's data access functionality	891

AAX_IACFPropertyMap	
Versioned interface for an AAX_IPropertyMap	893
AAX_IACFPropertyMap_V2	
Versioned interface for an AAX_IPropertyMap	895
AAX_IACFPropertyMap_V3	
Versioned interface for an AAX_IPropertyMap	898
AAX_IACFSessionDocument	
Interface representing information in a host session document	901
AAX_IACFSessionDocumentClient	
Interface representing a client of the session document interface	902
AAX_IACFTask	
Versioned interface for an asynchronous task	905
AAX_IACFTaskAgent	
Versioned interface for a component that accepts task requests	909
AAX_IACFTransport	
Versioned interface to get information about the host's transport state	912
AAX_IACFTransport_V2	
Versioned interface to get information about the host's transport state	917
AAX_IACFTransport_V3	
Versioned interface to get information about the host's transport state	921
AAX_IACFTransport_V4	
Versioned interface to get information about the host's transport state	924
AAX_IACFTransportControl	
Versioned interface to control the host's transport state	928
AAX_IACFViewContainer	
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the host app and by effect components	930
AAX_IACFViewContainer_V2	
Supplemental interface for the AAX host's view of a single instance of an effect. Used by both clients of the host app and by effect components	934
AAX_IACFViewContainer_V3	
Additional methods to track mouse as it moves over controls	937
AAX_IAutomationDelegate	
Interface allowing an AAX plug-in to interact with the host's event system	940
AAX_ICollection	
Interface to represent a plug-in binary's static description	946
AAX_IComponentDescriptor	
Description interface for an AAX plug-in component	951
AAX_IContainer	968
AAX_IController	
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAX host and by effect components	970
AAX_IDataBuffer	
Interface for reference counted data buffers	986
AAX_IDataBufferWrapper	
Wrapper for an AAX_IDataBuffer	988
AAX_IDescriptionHost	990
AAX_IDisplayDelegate< T >	991
AAX_IDisplayDelegateBase	
Defines the display behavior for a parameter	996
AAX_IDisplayDelegateDecorator< T >	
The base class for all concrete display delegate decorators	997
AAX_IDma	
Cross-platform interface for access to the host's direct memory access (DMA) facilities	1003
AAX_IEffectDescriptor	
Description interface for an effect's (plug-in type's) components	1014
AAX_IEffectDirectData	
The interface for a AAX Plug-in's direct data interface	1020

AAX_IEffectGUI	
The interface for a AAX Plug-in's user interface	1024
AAX_IEffectParameters	
The interface for an AAX Plug-in's data model	1027
AAX_IFeatureInfo	1033
AAX_IHostProcessor	
Base class for the host processor interface	1035
AAX_IHostProcessorDelegate	
Versioned interface for host methods specific to offline processing	1038
AAX_IHostServices	
Interface to diagnostic and debugging services provided by the AAX host	1041
AAX_IMIDIMessageInfoDelegate	1044
AAX_IMIDIINode	
Interface for accessing information in a MIDI node	1048
AAX_IPacketHandler	
Callback container used by AAX_CPacketDispatcher	1050
AAX_IPageTable	
Interface to the host's representation of a plug-in instance's page table	1051
AAX_IParameter	
The base interface for all normalizable plug-in parameters	1063
AAX_IParameterValue	
An abstract interface representing a parameter value of arbitrary type	1085
AAX_IPointerQueue< T >	1089
AAX_IPrivateDataAccess	
Interface to data access provided by host to plug-in	1092
AAX_IPropertyMap	
Generic plug-in description property map	1094
AAX_ISessionDocument	
Interface representing information in a host session document	1100
AAX_ISessionDocumentClient	
Interface representing a client of the session document interface	1102
AAX_IString	
A simple string container that can be passed across a binary boundary. This class, for simplicity, is not versioned and thus can never change	1104
AAX_ITaperDelegate< T >	1107
AAX_ITaperDelegateBase	
Defines the taper conversion behavior for a parameter	1111
AAX_ITask	
Interface representing a request to perform a task	1113
AAX_ITaskAgent	
Interface for a component that accepts task requests	1116
AAX_ITransport	
Interface to information about the host's transport state	1119
AAX_IViewContainer	
Interface for the AAX host's view of a single instance of an effect. Used both by clients of the AAX host and by effect components	1128
AAX_Map	1134
AAX_Point	
Data structure representing a two-dimensional coordinate point	1137
AAX_Rect	
Data structure representing a rectangle in a two-dimensional coordinate plane	1138
AAX_SHybridRenderInfo	
Hybrid render processing context	1140
AAX_SInstrumentPrivateData	
Utility struct for AAX_CMonolithicParameters	1141
AAX_SInstrumentRenderInfo	
Information used to parameterize AAX_CMonolithicParameters::RenderAudio()	1142

AAX_SInstrumentSetupInfo	
Information used to describe the instrument	1145
AAX_SPlugInChunk	
Plug-in chunk header + data	1153
AAX_SPlugInChunkHeader	
Plug-in chunk header	1156
AAX_SPlugInIdentifierTriad	
Plug-in Identifier Triad	1158
AAX_StLock_Guard	
Helper class for working with mutex	1159
AAX_TransportStateInfo_V1	1161
AAX_VAutomationDelegate	
Version-managed concrete automation delegate class	1163
AAX_VCollection	
Version-managed concrete AAX_ICollection class	1168
AAX_VComponentDescriptor	
Version-managed concrete AAX_IComponentDescriptor class	1175
AAX_VController	
Version-managed concrete Controller class	1190
AAX_VDataBufferWrapper	
Wrapper for an AAX_IDataBuffer	1207
AAX_VDescriptionHost	1210
AAX_VEffectDescriptor	
Version-managed concrete AAX_IEffectDescriptor class	1213
AAX_VFeatureInfo	1219
AAX_VHostProcessorDelegate	
Version-managed concrete Host Processor delegate class	1222
AAX_VHostServices	
Version-managed concrete AAX_IHostServices class	1225
AAX_VPageTable	
Version-managed concrete AAX_IPageTable class	1228
AAX_VPrivateDataAccess	
Version-managed concrete AAX_IPrivateDataAccess class	1241
AAX_VPropertyMap	
Version-managed concrete AAX_IPropertyMap class	1243
AAX_VSessionDocument	1250
AAX_VTask	
Version-managed concrete AAX_ITask	1252
AAX_VTransport	
Version-managed concrete AAX_ITransport class	1256
AAX_VViewContainer	
Version-managed concrete AAX_IViewContainer class	1267
AAX::Exception::Any	1273
AAX_CChunkDataParser::DataValue	1277
IACFDefinition	
Publicly inherits from IACFUnknown . This abstract interface is used to identify all of the plug-in components in the host	1279
IACFUnknown	
COM compatible IUnknown C++ interface	1282
AAX::Exception::ResultError	1284
SAutoArray< T >	1287
AAX_ISessionDocument::TempoMap	1288
AAX_VSessionDocument::VTempoMap	1290

Chapter 11

File Index

11.1 File List

Here is a list of all files with brief descriptions:

AAX_ACFInterface.doxygen	1293
AAX_AdditionalFeatures_Algorithm.doxygen	1294
AAX_AdditionalFeatures_AOSandSidechain.doxygen	1294
AAX_AdditionalFeatures_CurveDisplays.doxygen	1294
AAX_AdditionalFeatures_Hybrid.doxygen	1294
AAX_AdditionalFeatures_Meters.doxygen	1294
AAX_AdditionalFeatures_MIDI.doxygen	1294
AAX_AuxInterface_DirectData.doxygen	1294
AAX_AuxInterface_HostProcessor.doxygen	1294
AAX_AuxInterface_TaskAgent.doxygen	1294
AAX_BugList.doxygen	1294
AAX_CommonInterface_Algorithm.doxygen	1294
AAX_CommonInterface_Communication.doxygen	1294
AAX_CommonInterface_DataModel.doxygen	1294
AAX_CommonInterface_Describe.doxygen	1294
AAX_CommonInterface_FormatSpecification.doxygen	1295
AAX_CommonInterface_GUI.doxygen	1295
AAX_DigiTrace_Guide.doxygen	1295
AAX_DistributingYourPlugIn.doxygen	1295
AAX_DocsDirectory.doxygen	1295
AAX_Getting_Started_Guide.doxygen	1295
AAX_HostSupport.doxygen	1295
AAX_InstrumentParameters.doxygen	1295
AAX_InterfaceList.doxygen	1295
AAX_LinkedParameters.doxygen	1295
AAX_Media_Composer_Guide.doxygen	1295
AAX_OtherExtensions.doxygen	1295
AAX_Page_Table_Guide.doxygen	1295
AAX_ParameterAutomation.doxygen	1295
AAX_ParameterManager.doxygen	1295
AAX_ParameterUpdateProtocol.doxygen	1295
AAX_ParameterUpdateTiming.doxygen	1295
AAX_Pro_Tools_Guide.doxygen	1296
AAX_RealTimePerformance.doxygen	1296
AAX_RelatedTypes.doxygen	1296

AAX_SDK_ChangeLog.doxygen	1296
AAX_SDK_ExamplePlugIns.doxygen	1296
AAX_SDK_GUIExtensions.doxygen	1296
AAX_TI_Guide.doxygen	1296
AAX_Troubleshooting.doxygen	1296
AAX_VENUE_Guide.doxygen	1296
DSH_Guide.doxygen	1296
DTT_Guide.doxygen	1296
ReadMe.doxygen	1296
AAX_MIDILogging.cpp	1296
AAX_MIDILogging.h	
Utilities for logging MIDI data	1297
AAX_PluginBundleLocation.h	
Utilities for interacting with the host OS	1297
AAX_CMonolithicParameters.cpp	1298
AAX_CMonolithicParameters.h	
A convenience class extending AAX_CEffectParameters for monolithic instruments	1298
AAX.h	
Various utility definitions for AAX	1303
AAX_Assert.h	
Declarations for cross-platform AAX_ASSERT, AAX_TRACE and related facilities	1323
AAX_Atomic.h	
Atomic operation utilities	1331
AAX_Callbacks.h	
AAX callback prototypes and ProcPtr definitions	1339
AAX_CArrayDataBuffer.h	1343
AAX_CAtomicQueue.h	
Atomic, non-blocking queue	1346
AAX_CAutoreleasePool.h	
Autorelease pool helper utility	1350
AAX_CBinaryDisplayDelegate.h	
A binary display delegate	1351
AAX_CBinaryTaperDelegate.h	
A binary taper delegate	1354
AAX_CChunkDataParser.h	
Parser utility for plugin chunks	1355
AAX_CDecibelDisplayDelegateDecorator.h	
A decibel display delegate	1358
AAX_CEffectDirectData.h	
A default implementation of the AAX_IEffectDirectData interface	1360
AAX_CEffectGUI.h	
A default implementation of the AAX_IEffectGUI interface	1362
AAX_CEffectParameters.h	
A default implementation of the AAX_IeffectParameters interface	1363
AAX_CHostProcessor.h	
Concrete implementation of the AAX_IHostProcessor interface for non-real-time processing	1367
AAX_CHostServices.h	
Concrete implementation of the AAX_IHostServices interface	1369
AAX_CLinearTaperDelegate.h	
A linear taper delegate	1370
AAX_CLogTaperDelegate.h	
A log taper delegate	1372
AAX_CMutex.h	
Mutex	1373
AAX_CNumberDisplayDelegate.h	
A number display delegate	1374
AAX_CommonConversions.h	1376

AAX_CPacketDispatcher.h	
Helper classes related to posting AAX packets and handling parameter update events	1384
AAX_CParameter.h	
Generic implementation of an AAX_IParameter	1387
AAX_CParameterManager.h	
A container object for plug-in parameters	1400
AAX_CPercentDisplayDelegateDecorator.h	
A percent display delegate decorator	1401
AAX_CPieceWiseLinearTaperDelegate.h	
A piece-wise linear taper delegate	1403
AAX_CRangeTaperDelegate.h	
A range taper delegate decorator	1407
AAX_CSessionDocumentClient.h	
A state display delegate	1410
AAX_CStateDisplayDelegate.h	
A state display delegate	1411
AAX_CStateTaperDelegate.h	
A state taper delegate (similar to a linear taper delegate.)	1414
AAX_CString.h	
A generic AAX string class with similar functionality to std::string	1416
AAX_CStringDataBuffer.h	
A string display delegate	1420
AAX_CStringDisplayDelegate.h	
A string display delegate	1422
AAX_CTaskAgent.h	
A default implementation of the AAX_ITaskAgent interface	1424
AAX_CUnitDisplayDelegateDecorator.h	
A unit display delegate decorator	1425
AAX_CUnitPrefixDisplayDelegateDecorator.h	
A unit prefix display delegate decorator	1427
AAX_EndianSwap.h	
Utility functions for byte-swapping. Used by AAX_CChunkDataParser	1429
AAX_Enums.h	
Utility functions for byte-swapping. Used by AAX_CChunkDataParser	1437
AAX_EnvironmentUtilities.h	
Useful environment definitions for AAX	1489
AAX_Errors.h	
Definitions of error codes used by AAX plug-ins	1490
AAX_Exception.h	
AAX SDK exception classes and utilities	1503
AAX_Exports.cpp	
Constants and other definitions used by AAX plug-in GUIs	1510
AAX_GUITypes.h	
Constants and other definitions used by AAX plug-in GUIs	1514
AAX_IACFAutomationDelegate.h	
Versioned interface allowing an AAX plug-in to interact with the host's automation system	1519
AAX_IACFCollection.h	
Versioned interface to represent a plug-in binary's static description	1521
AAX_IACFComponentDescriptor.h	
Versioned description interface for an AAX plug-in algorithm callback	1522
AAX_IACFController.h	
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components	1523
AAX_IACFDataBuffer.h	
The direct data access interface that gets exposed to the host application	1526
AAX_IACFDescriptionHost.h	
The GUI interface that gets exposed to the host application	1527
AAX_IACFEffectDescriptor.h	
Versioned interface for an AAX_IEffectDescriptor	1528
AAX_IACFEffectDirectData.h	
The direct data access interface that gets exposed to the host application	1529
AAX_IACFEffectGUI.h	
The GUI interface that gets exposed to the host application	1530

AAX_IACFEffEffectParameters.h	
The data model interface that is exposed to the host application	1531
AAX_IACFFeatureInfo.h	1534
AAX_IACFHostProcessor.h	
The host processor interface that is exposed to the host application	1535
AAX_IACFHostProcessorDelegate.h	1536
AAX_IACFHostServices.h	1537
AAX_IACFPageTable.h	1538
AAX_IACFPageTableController.h	1539
AAX_IACFPrivateDataAccess.h	
Interface for the AAX host's data access functionality	1541
AAX_IACFPropertyMap.h	
Versioned interface for an AAX_IPropertyMap	1542
AAX_IACFSessionDocument.h	1543
AAX_IACFSessionDocumentClient.h	1544
AAX_IACFTask.h	
Defines the interface representing an asynchronous task	1545
AAX_IACFTaskAgent.h	1547
AAX_IACFTransport.h	
Interface for accessing the host's transport state	1548
AAX_IACFTransportControl.h	
Interface for control over the host's transport state	1550
AAX_IACFViewContainer.h	
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components	1551
AAX_IAutomationDelegate.h	
Interface allowing an AAX plug-in to interact with the host's automation system	1552
AAX_ICollection.h	
Interface to represent a plug-in binary's static description	1553
AAX_IComponentDescriptor.h	
Description interface for an AAX plug-in algorithm	1554
AAX_IContainer.h	
Abstract container interface	1556
AAX_IController.h	
Interface for the AAX host's view of a single instance of an effect	1557
AAX_IDataBuffer.h	1559
AAX_IDataBufferWrapper.h	1560
AAX_IDescriptionHost.h	1561
AAX_IDisplayDelegate.h	
Defines the display behavior for a parameter	1562
AAX_IDisplayDelegateDecorator.h	
The base class for all concrete display delegate decorators	1563
AAX_IDma.h	
Cross-platform interface for access to the host's direct memory access (DMA) facilities	1565
AAX_IEffectDescriptor.h	
Description interface for an effect's (plug-in type's) components	1567
AAX_IEffectDirectData.h	
Optional interface for direct access to alg memory	1568
AAX_IEffectGUI.h	
The interface for a AAX Plug-in's user interface	1569
AAX_IEffectParameters.h	
The interface for an AAX Plug-in's data model	1570
AAX_IFeatureInfo.h	1570
AAX_IHostProcessor.h	
Base class for the host processor interface which is extended by plugin code	1571
AAX_IHostProcessorDelegate.h	
Interface allowing plug-in's HostProcessor to interact with the host's side	1572

AAX_IHostServices.h	
Various host services	1573
AAX_IMIDINode.h	
Declaration of the base MIDI Node interface	1574
AAX_Init.h	
AAX library implementations of required plug-in initialization, registration, and tear-down methods	1575
AAX_IPageTable.h	1579
AAX_IParameter.h	
The base interface for all normalizable plug-in parameters	1580
AAX_IPointerQueue.h	
Abstract interface for a basic FIFO queue of pointers-to-objects	1583
AAX_IPrivateDataAccess.h	
Interface to data access provided by host to plug-in	1584
AAX_IPropertyMap.h	
Generic plug-in description property map	1585
AAX_ISessionDocument.h	1586
AAX_ISessionDocumentClient.h	1587
AAX_IString.h	
An AAX string interface	1588
AAX_ITaperDelegate.h	
Defines the taper conversion behavior for a parameter	1589
AAX_ITask.h	1590
AAX_ITaskAgent.h	1591
AAX_ITransport.h	
The interface for query ProTools transport information	1592
AAX_IViewContainer.h	
Interface for the AAX host's view of a single instance of an effect	1593
AAX_MIDIUtilities.h	
Utilities for managing MIDI data	1595
AAX_PageTableUtilities.h	1598
AAX_PopStructAlignment.h	
Resets (pops) the struct alignment to its previous value	1601
AAX_PostStructAlignmentHelper.h	
Helper file for data alignment macros	1602
AAX_PreStructAlignmentHelper.h	
Helper file for data alignment macros	1602
AAX_Properties.h	
Contains IDs for properties that can be added to an AAX_IPropertyMap	1603
AAX_Push2ByteStructAlignment.h	
Set the struct alignment to 2-byte. This file will throw an error on platforms that do not support 2-byte alignment (i.e. TI DSPs)	1626
AAX_Push4ByteStructAlignment.h	
Set the struct alignment to 4-byte	1627
AAX_Push8ByteStructAlignment.h	
Set the struct alignment to 8-byte	1628
AAX_SessionDocumentTypes.h	1630
AAX_SliderConversions.h	
Legacy utilities for converting parameter values to and from the normalized full-scale 32-bit fixed domain that was used for RTAS/TDM plug-ins	1631
AAX_StringUtilities.h	
Various string utility definitions for AAX Native	1635
AAX_StringUtilities.hpp	1637
AAX_TransportTypes.h	
Structures, enums and other definitions used in transport	1649
AAX_UIDs.h	
Unique identifiers for AAX/ACF interfaces	1651

AAX_UtilsNative.h	Various utility definitions for AAX Native	1671
AAX_VAutomationDelegate.h	Version-managed concrete AutomationDelegate class	1673
AAX_VCollection.h	Version-managed concrete Collection class	1674
AAX_VComponentDescriptor.h	Version-managed concrete ComponentDescriptor class	1675
AAX_VController.h	Version-managed concrete Controller class	1677
AAX_VDataBufferWrapper.h	1679
AAX_VDescriptionHost.h	1680
AAX_VEffectDescriptor.h	Version-managed concrete EffectDescriptor class	1681
AAX_Version.h	Version stamp header for the AAX SDK	1682
AAX_VFeatureInfo.h	1687
AAX_VHostProcessorDelegate.h	Version-managed concrete HostProcessorDelegate class	1688
AAX_VHostServices.h	Version-managed concrete HostServices class	1689
AAX_VPageTable.h	1690
AAX_VPrivateDataAccess.h	Version-managed concrete PrivateDataAccess class	1692
AAX_VPropertyMap.h	Version-managed concrete PropertyMap class	1693
AAX_VSessionDocument.h	1694
AAX_VTask.h	1695
AAX_VTransport.h	Version-managed concrete Transport class	1696
AAX_VViewContainer.h	Version-managed concrete ViewContainer class	1698
AAX_Alignment.h	Alignment malloc and free methods for optimization	1699
AAX_Constants.h	Signal processing constants	1700
AAX_Denormal.h	Signal processing utilities for denormal/subnormal floating point numbers	1703
AAX_FastInterpolatedTableLookup.h	A set of functions that provide lookup table functionality. Not necessarily optimized for TI, but used internally	1707
AAX_FastInterpolatedTableLookup.hpp	1709
AAX_FastPow.h	Set of functions to optimize pow	1710
AAX_Map.h	1713
AAX_MiscUtils.h	Miscellaneous signal processing utilities	1714
AAX_PlatformOptimizationConstants.h	Constants descriptor..	1720
AAX_Quantize.h	Quantization utilities	1721
AAX_RandomGen.h	Functions for calculating pseudo-random numbers	1723
AAX_SampleRateUtils.h	Description	1725

Chapter 12

Module Documentation

12.1 AAX SDK Manual

12.1.1

12.1.2 Welcome to AAX

Select the "Manual" tab to see a full list of documentation pages, or choose from the topics below.

Note

Looking for something? The search function only includes indexing of code symbols and page titles. To search for specific text strings in the AAX SDK manual it is best to use a text search tool such as grep or FINDSTR on the AAX SDK directory or search for the desired text within the PDF version of the AAX SDK documentation.

12.1.2.1 The Basics

New to AAX? Read through the documentation pages listed below to get started!

- See [Getting Started with AAX](#) for a general overview of AAX and a walk-through of the DemoGain example plug-in
- Read through the first few sections of the [Pro Tools Guide](#) if you are new to Pro Tools
- Read the [Digital signature](#) section of the [Pro Tools Guide](#) to review the digital signing requirements for compatibility with Pro Tools
- Review the [sample plug-ins](#) for examples of both basic and advanced AAX features
- See [Core AAX Interface](#) to find out more about the basic structure of AAX plug-ins
- See [Real-time algorithm callback](#) to learn more about audio processing in AAX
- See [Data model interface](#) to learn about adding parameters and controls to your plug-in,
- See [Description callback](#) for more information about plug-in configuration and initial set-up
- See the [HDX DSP Guide](#) to find out how to add AAX DSP support to your plug-in for Avid's HDX and Pro Tools | Carbon products
- Ready to ship your new plug-in? See [Distributing Your AAX Plug-In](#) for information about finalizing and distributing your AAX products
- Check out the [Troubleshooting](#) page if you're having problems, or post a question to the [developer forums](#) and an Avid engineer will be happy to assist you.

12.1.2.2 More Topics

Have a more specific question? Review the pages below or view the full list of documentation pages under the "Manual" tab above.

- See [AAX_IController](#) to see the interface that an AAX plug-in's host-based modules use to interact with the host
- See [AAX communication protocols](#) to find out more about how different modules in an AAX plug-in communicate with one another
- See [Offline processing interface](#) for information about creating advanced non-real-time AAX plug-ins
- See [Taper delegates](#) and [Display delegates](#) for more information about implementing custom parameter and control behavior
- Look in the /TI/SignalProcessing folder for signal processing utility classes and functions available for optimizing on Native and DSP

12.1.2.3 Test Tools & Utilities

- The [DSH Guide](#) has information about the tool for testing basic functionality of your plug-in
- See [DTT Guide](#) to learn how to automate different test scenarios for DSH

12.1.2.4 Supplemental Information

- [Example Plug-Ins](#)
- [Change Log](#)
- [Host Support](#)
- [Known Issues](#)

12.1.3 SDK Folder Hierarchy

Documentation

SDK documentation

ExamplePlugIns

Example plug-in projects [More information](#)

Extensions

Demonstrations of how to extend the AAX SDK, for example by incorporating third-party GUI frameworks into AAX plug-ins. [More information](#)

Interfaces

Interface headers and other resources required for use of the AAX SDK library

Libs

Source code for the AAX SDK library, a collection of default implementations and utility classes for use in all AAX plug-ins

TI

Various resources for use with TI's Code Composer Studio IDE and Avid's TI testing toolset

Utilities

Common SDK utilities and resources

12.1.4 Contacting Avid

Your personal [avid user account](#) is your hub for AAX Toolkit services and developer support.

Log in at [avid.com](#) for access to the full range of tools and services provided to AAX developers, including the AAX [developer forum](#). If you have any questions on the AAX SDK documentation or require support with AAX development, we encourage you to post them to the forum as your first line of inquiry.

If you have time-sensitive or critical support inquiries, contact the AAX development team directly at devservices@avid.com. Any AAX questions sent to this alias will be promptly addressed by the most appropriate contact here at Avid.

If you require NFR (Not For Resale) licenses to Avid software for AAX development please send an e-mail to devauth@avid.com with "License Request" in the subject.

If you require access to the digital signing toolkit from PACE Anti-Piracy, Inc. for compatibility with Pro Tools then please follow the instructions [here](#).

The following chart describes these and other ways of connecting with Avid to take advantage of the services provided to AAX developers:

12.1.5 Licensing

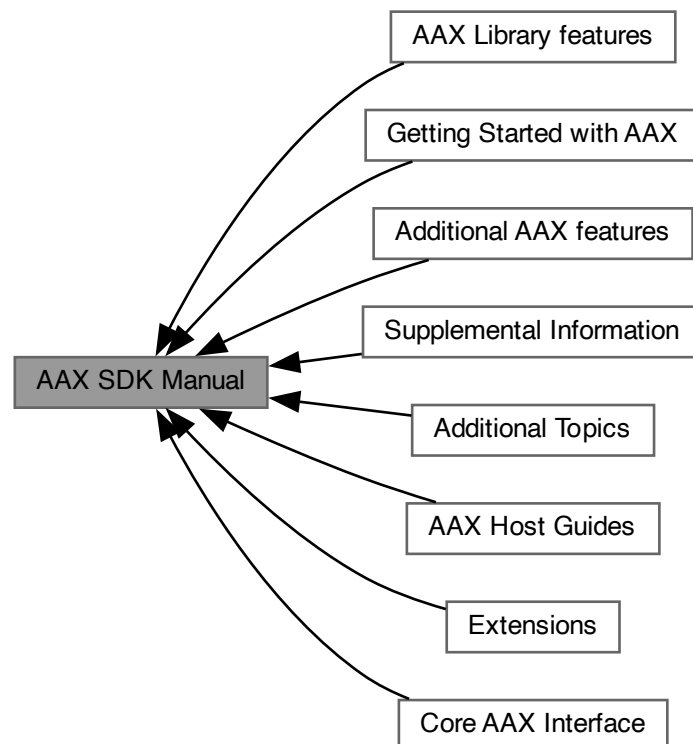
Unless you have entered into a commercial agreement with Avid, you are using this SDK under an evaluation agreement. To review this agreement, see the [AAX Toolkit downloads](#) section under your [my.avid.com](#) account.

As an Avid Developer, you are invited to offer your products on [Avid Marketplace](#) and via [Avid Link](#). If you wish to sell them independently or through other commercial outlets, an authorized representative from your organization is required to sign our Commercial License, which you can read and click through [here](#).

Documents

- [Getting Started with AAX](#)
A brief introduction to AAX.
- [Core AAX Interface](#)
Main classes, callbacks, and format specification details for a standard AAX plug-in.
- [Additional AAX features](#)
How to use additional features and functionality supported by AAX.
- [AAX Library features](#)
AAX Library core support for the AAX interface
- [Additional Topics](#)
Additional information about the AAX design.
- [AAX Host Guides](#)
Documentation for specific AAX host environments.
- [Extensions](#)
Extensions to the AAX SDK.
- [Supplemental Information](#)
Supplemental documents beyond the scope of the AAX SDK.

Collaboration diagram for AAX SDK Manual:



12.2 Getting Started with AAX

A brief introduction to AAX.

12.2.1 Contents

- [Welcome](#)
- [Quick Start](#)
- [AAX Design Overview](#)
- [DemoGain Example](#)
- [Next Steps](#)

12.2.2 Welcome

Welcome to AAX! This guide is designed to introduce you to the fundamental concepts of AAX. By the end of this guide you will understand:

- The purpose of AAX
- The basic components of an AAX plug-in
- The structure of the DemoGain example plug-in
- What you need to do to successfully build and run your own AAX plug-ins

12.2.3 Quick Start

Use the steps below to get up and running quickly with an example plug-in from the AAX SDK.

- Build the AAX Library

If this is your first time opening the AAX SDK then your first step should be to build the AAX Library project. The AAX Library is a static library containing default implementations of the AAX interface and convenience classes designed to make AAX development easy. All of the SDK example plug-ins link to this library, and your plug-ins should too.

Open the AAX Library project for your chosen IDE from the Libs/AAXLibrary directory and build the library. Now you are ready to build plug-ins!

- Open and build the DemoGain example plug-in

The DemoGain project is located in ExamplePlugIns/DemoGain. Once you have built the AAX Library project you will be able to successfully build DemoGain. To learn more about the DemoGain example plug-in, see the [DemoGain Example](#) section below.

- Install a developer build of Pro Tools

If you looking at AAX then you are most likely interested in developing plug-ins for [Pro Tools](#). Publicly available builds of Pro Tools require that AAX plug-ins be digitally signed. During development, you can use a "developer build" of Pro Tools to run unsigned test builds of your plug-ins, like the DemoGain plug-in that you just built.

Download and install the latest Pro Tools developer build from the AAX developer downloads in your [my.↔
avid.com](#) account.

- Obtain a Pro Tools Developer iLok license (and an iLok)

Pro Tools developer builds require a special license loaded on an iLok USB device. You can request this Pro Tools Developer license, as well as any other NFR (Not For Resale) licenses which you require for your AAX product development and testing, by writing to devauth@avid.com with "License Request" in the subject.

If you don't have an iLok device you will also need to obtain one from [PACE Anti-Piracy Inc.](#) or a reseller, including most music shops which sell audio software.

- Install DemoGain in the AAX Plug-Ins folder

In order to be loaded into Pro Tools, a plug-in must be present in the system's AAX Plug-Ins directory. See [Install directories](#) in the [Pro Tools Guide](#) document for more information about where to install AAX plug-ins for Pro Tools.

- Launch Pro Tools and run DemoGain

You're now ready to run your very first AAX plug-in!

1. Make sure that your iLok USB device is connected to the system and contains the Pro Tools Developer license, then launch the Pro Tools developer build.
2. Once Pro Tools has launched, you will be prompted to create a new Session or Project. Choose Session (Local Storage) to create a local session file.
3. Create a new mono audio track in your session using the "New Track..." menu item
4. If it is not already visible, reveal the "Inserts A-E" view in the Edit window using the View > Edit Window menu.
5. Click on one of the insert slots for your audio track and choose DemoGain from the insert selection menu.

You're now running an instance of the DemoGain AAX plug-in. If you have a debugger attached to Pro Tools you should now be able to break in the plug-in's methods and step through its code.

- Get ready to release your own products

Once you have created your own AAX plug-in you can follow the steps in [Distributing Your AAX Plug-In](#) to prepare your plug-in for public sale and distribution.

In particular, for Pro Tools compatibility or to test your product with a public Pro Tools release you will need to digitally sign your plug-in using a toolkit from PACE Anti-Piracy Inc. See the [digital signature](#) section of the [Pro Tools Guide](#) for more information about this requirement.

12.2.4 AAX Design Overview

12.2.4.1 Architecture Philosophy

The purpose of the AAX architecture is to provide an *easy-to-use* framework for the development of *high-performance audio plug-ins* that can run on a *variety of platforms*, both present and future, with a *high level of code re-use*.

12.2.4.2 Design Attributes

AAX incorporates a flexible, decoupled component hierarchy, including:

1. A relocatable C-style callback that performs the plug-in's real-time audio processing algorithm
2. A collection of supporting C++ objects that manage the plug-in's data, GUI, and any other non-real-time information

3. A "description" method that statically describes the plug-in's layout and compatibility requirements to the host.

This flexible design facilitates optimal performance and portability; AAX is 64-bit ready and is designed to work well in both host-based (Native), accelerated (DSP), and future (e.g. embedded) environments. Importantly, plug-ins running in each of these environments can use the same code base, and porting to new platforms should not require much more than a re-compile. To satisfy these portability requirements, AAX must be decoupled into three parts, the GUI, data model, and algorithm.

12.2.4.3 Component Structure

The core component structure of an AAX plug-in involves data model, GUI, description, and algorithm modules. The design involves a mixture of C++ objects (data model and GUI modules) and C-style callbacks (algorithm and description modules.)

The figure below shows basic object ownership and composition in an AAX plug-in. The purple components are provided as part of the AAX SDK while the other components are written as needed by the plug-in developer. The classes above the dotted line are pure virtual interfaces, while the classes below the dotted line are concrete implementations.

Figure 1: Core AAX interface design.

As you can see, the plug-in's [data model](#) and [GUI](#) are written as C++ objects inherited from SDK interfaces, while its [algorithm](#) and [Description](#) methods are implemented as simple callbacks. This basic model may be expanded by attaching additional modules, such as the [Host Processor](#) module used by advanced non-real-time plug-ins. In this section, however, we will be considering only the four core interfaces and modules.

12.2.4.4 Algorithm

The most fundamental, and most important, component of any audio plug-in is its processing algorithm. Due to the design requirements of an AAX plug-in, this component must meet several constraints in order to be compatible with accelerated platforms:

1. It must be possible to build the algorithm as a compatible component on a variety of platforms
2. It must be possible for the host to re-locate the algorithm in memory without affecting the algorithm's state
3. The algorithm must be separable from the rest of its plug-in, e.g. into a different memory space
4. The algorithm must be as efficient as possible to call.

To satisfy these constraints, AAX uses a decoupled C-style callback function. State information within the function is obtained through the context, a custom data structure that contains everything from the "outside world" that is relevant to the algorithm. The context includes information such as audio buffers and coefficient packets, which are provided according to a static set of data routing definitions that we will describe further in the next chapter. The host is responsible for ensuring that this structure is up-to-date whenever the algorithm callback is entered.

See also

[Real-time algorithm callback](#)

12.2.4.5 Data Model

The [AAX_IEffectParameters](#) interface represents the data model portion of your plug-in. The AAX host interacts with your plug-in's data model via this interface, which includes methods that store and update of your plug-in's internal data.

In your plug-in's data model implementation, you will be responsible for creating the plug-in's parameters and defining how the plug-in will respond when these parameters are changed via control updates or preset loads. In order for information to be routed from the data model to the plug-in's algorithm, the parameters that are created here must be registered with the host in the plug-in's Description callback (see below).

The data model is composed with [AAX_IController](#), an interface that provides access to the host. This interface provides a means of querying information from the host such as stem format or sample rate, and is also responsible for communication between the data model and the (decoupled) algorithm.

You will most likely inherit your custom data model's implementation from [AAX_CEffectParameters](#), a default implementation of the [AAX_IEffectParameters](#) interface. This class provides basic functionality such as adding custom parameters, setting control values, restoring state, generating coefficients, etc., which you can override and customize as needed.

See also

[Data model interface](#)

12.2.4.6 GUI Interface

The [AAX_IEffectGUI](#) interface contains the plug-in's GUI methods. This interface is decoupled from the plug-in's data model, allowing AAX to support distributed architectures that incorporate remote GUIs.

The GUI is also composed with [AAX_IController](#), from which references to the plug-in's other components, such as its data model interface, may be retrieved.

The AAX SDK includes a set of GUI extensions to help you get started implementing your plug-ins' GUIs. These extensions include both native drawing formats and suggested integrations with third-party graphics frameworks. Although the SDK does not include any actual graphics frameworks, these extensions provide examples of how you can incorporate your chosen GUI framework with [AAX](#).

See also

[GUI interface](#)

12.2.4.7 Describe

A plug-in's Describe callback ties all the plug-in's components together and registers the plug-in with the host. In this callback, your plug-in defines a set of algorithm callbacks, data connections, and static plug-in properties

In order to route data to the plug-in's algorithm, Describe includes a description of the algorithm's context structure. This description involves a set of port definitions, which can be "hard-wired" to receive data from the host (such as audio buffers,) from the plug-in's data model (such as packets of coefficients,) or even from past calls to the algorithm (private, persistent algorithm state.)

Once these connections are made, Describe passes the host a populated description interface and returns. In the next section we will demonstrate how all these interfaces interact with one another by examining a sample plug-in.

See also

[Description callback](#)

12.2.4.8 Controller

There is one additional core component to any AAX plug-in, the Controller. The plug-in does not implement this component - rather, the Controller is an interface that provides access to various facilities provided by the host, as well as to the plug-in's other modules.

12.2.5 DemoGain Example

The AAX SDK includes a basic example plug-in named DemoGain. In this section we will examine this plug-in to show how the various core modules in an AAX plug-in interact with one another. We will focus in particular on how the DemoGain's "gain" parameter is defined, routed to the algorithm, and used to apply a gain effect to audio samples. In this way we will "visit" each of the core components in DemoGain.

For a description of all the example plug-ins included in the SDK, see the [Example Plug-Ins](#) page.

Note

To run DemoGain or other example plug-ins in Pro Tools 10 you must change the `AAX_ePlugInCategory_Example` category to `AAX_ePlugInCategory_Dynamics` in the plug-in's Describe function. The "example" category is not supported in Pro Tools 10.

12.2.5.1 AAX Plug-In Parameters

A good starting point for understanding a plug-in is to understand its parameters. In DemoGain, as in most AAX plug-ins, the plug-in's parameters define its data model state and the plug-in's parameters provide the fundamental connection between user interactions and audio processing.

The sections below will guide you through each of the main steps of parameter creation and connection, making use of the default implementation in [AAX_CEffectParameters](#):

1. [Data Model: Create Your Parameter](#)
 - (a) Create a new parameter object
 - (b) Register the parameter with the Packet Dispatcher
 - (c) Create an update callback that converts the raw parameter value into a packet of processed coefficients
2. [Algorithm: Add coefficients to the algorithm's context structure](#)
 - (a) Create a new field for incoming coefficient packets
 - (b) Generate a *field ID* to reference the new member
 - (c) Add the new coefficient to the plug-in's algorithm code
3. [Describe: Connect the parameter throughout the plug-in](#)
 - (a) Add a new Data Input Port to the algorithm via the component descriptor interface
 - (b) Add a connection between the parameter's *packet ID* and its coefficients' *field ID*
4. [GUI: Add a control](#)
 - (a) Create a GUI widget that can update the parameter's state
 - (b) Add logic to notify the data model when the GUI is edited
 - (c) Add logic to update the GUI when the data model state changes

12.2.5.2 Data Model: Create Your Parameter

DemoGain's data model inherits its functionality from [AAX_CEffectParameters](#) (the default implementation of [AAX_IEffectParameters](#)). The `DemoGain_Parameters.h` and `DemoGain_Parameters.cpp` source files comprise the source code for DemoGain's data model.

During plug-in initialization, `DemoGain_Parameters::EffectInit()` creates the plug-in's custom parameters. A look inside of the `.cpp` file shows how this is done via the creation of a new [AAX_CParameter](#) for the gain parameter: the [AAX_CParameter](#) constructor is parametrized with a set of arguments that define the gain parameter's behavior, such as its default value (1.0f), name ("Gain"), taper behavior (linear), etc.

After creating the parameter objects, a series of packets are registered with the host via the inherited [Packet Dispatcher](#), `mPacketDispatcher`, which is a helper object used by [AAX_CEffectParameters](#) to assist with the plug-in's packet management tasks.

```
mPacketDispatcher.RegisterPacket(
    DemoGain_GainID,
    eAlgPortID_CoefsGainIn,
    this,
    &DemoGain_Parameters::UpdatePacket_Gain);
```

Listing 1: Registration of DemoGain's "gain" packet handler

The last parameter in [AAX_CPacketDispatcher::RegisterPacket\(\)](#) takes a reference to a packet handler callback. This method will be called by the Packet Dispatcher whenever new parameter values need to be converted into coefficients that can be used by the algorithm. In DemoGain, a reference to `DemoGain_Parameters::UpdatePacket_Gain()` is used for the gain parameter's coefficient packet.

As a developer, you will choose how this portion of your data model gets handled; you can choose to use the default handler method, which simply forwards the raw parameter values to the algorithm, or you can define a custom handler. You can also choose to bypass the Packet Dispatcher completely (see the `DemoDist_GenCoef` plug-in for an example of this.)

Although the handling of DemoGain's gain parameter is trivial, for the sake of demonstration this plug-in uses both Packet Dispatcher approaches in the registration of its bypass and gain parameters.

12.2.5.3 Algorithm: Add coefficients to the algorithm's context structure

Your plug-in's context is nothing more than a data structure of pointers that is registered with the host during Describe. These pointers function as *ports* where host-managed data may be delivered or retrieved.

12.2.5.3.1 Context definition Looking under the "Component context definitions" section within `DemoGain_Alg.h`, you will find DemoGain's `SDemoGain_Alg_Context` context. This is DemoGain's context structure, and its sole purpose is to parametrize DemoGain's algorithm with a set of ports. Note the definition of a port for the gain coefficients that are created by `DemoGain_Parameters::UpdatePacket_Gain()` and another port for the bypass value that is forwarded via the default packet update handler.

```
int32_t          * mCtrlBypassP;
SDemoGain_CoefsGain * mCoefsGainP;
```

Listing 2: Gain and bypass ports in the DemoGain algorithm's context structure

Once ports have been defined for the algorithm's coefficients and other algorithmic data, unique identifiers for each port in the context must be generated. This is accomplished through use of the [AAX_FIELD_INDEX](#) macro. The basic idea behind this macro is to generate IDs that the host can use to directly address the ports within their context:

```
, eAlgPortID_CoefsGainIn = AAX_FIELD_INDEX ( SDemoGain_Alg_Context , mCoef ),
, eAlgFieldID_AudioIn = AAX_FIELD_INDEX ( SDemoGain_Alg_Context , mInputPP ) )
```

Listing 3: Some port ID definitions for the DemoGain algorithm's context structure

Now the context's definition is complete. So far these are just fields in a struct; the host doesn't yet know how to route packets from the data model to these ports. That information will come later, in the plug-in's [description](#). Once this context is described to the host, the host will know to populate it and pass it to the processing function located in `DemoGain_AlgProc.cpp`.

12.2.5.3.2 Algorithm processing callback

```
void
AAX_CALLBACK
DemoGain_AlgorithmProcessFunction (
    SDemoGain_Alg_Context * const inInstancesBegin [],
    const void * inInstancesEnd )
```

Listing 4: The DemoGain algorithm's callback prototype

This is where the plug-in's audio buffer processing of the plug-in occurs. Note that this function is passed a pointer to an array of `SDemoGain_Alg_Contexts`. Each of these represents the state of a particular instance of DemoGain, and contains pointers to the applicable coefficient and audio buffer data.

Using the `SDemoGain_Alg_Context` array, instance-specific information and audio buffers are easily retrieved for processing. DemoGain accomplishes this by first iterating over each plug-in instance, then over each channel, and finally over each sample, at which point the gain coefficient is applied to each sample in the input audio buffer and the sample data is copied to the output audio buffer:

```
// ----- Iterate over plug-in instances -----//
for (SDemoGain_Alg_Context * const * walk = inInstancesBegin ; walk &lt; inInstancesEnd ; ++ walk )
{
    .
    .
    .
    // ----- Run processing loop over each input channel -----//
    //
    for (int ch = 0; ch &lt; kNumChannelsIn ; ch++)
    {
        // ----- Run processing loop over each sample -----//
        //
        for (int t = 0; t &lt; kAudioWindowSize ; t++)
        {
            .
            .
            .
            pdO [t] = gain * pdI [t];
            .
            .
            .
        } // Go to the next sample
    } // Go to next channel
} // End instance-iteration loop
```

Listing 5: Iterative loops in the DemoGain algorithm

12.2.5.4 Describe: Connect the parameter throughout the plug-in

As mentioned before, Describe provides a static description of all communication pathways between a plug-in's algorithm, host, and data model. In addition, various effect properties are defined that help the host determine how to handle the plug-in.

Communication paths between the various plug-in components are described as connections between source and destination ports. In order for these communication paths to be created, the algorithm must first define some destination ports by actually registering its previously defined context fields as communication destinations. DemoGain does this for its gain parameter through the following call to `AAX_IComponentDescriptor::AddDataInPort()` in `DemoGain_Describe.cpp`'s `DescribeAlgorithmComponent()` method:

```
AAX_IComponentDescriptor * compDesc;
compDesc = outDescriptor->NewComponentDescriptor ();
err = compDesc->AddDataInPort (
    eAlgPortID_CoefsGainIn ,
    sizeof (SDemoGain_CoefsGain) );
```

Listing 6: Adding a data port to DemoGain's algorithm component descriptor

This registration process is required for both custom coefficients (Gain) and for data that all plug-ins need such as audio input and output fields.

That completes the connection, and now the plug-in is fully wired to receive parameter updates, convert raw parameter values to algorithmic coefficients, pack these coefficients into a packet, post the packet to the host for routing, receive the updated packet in the list of context structures that the host provides when calling the algorithm callback, and apply the updated coefficient data in the appropriate context structure to the plug-in's audio data. Whew!

12.2.5.5 GUI: Add a control

Although the specific steps for adding a GUI control to edit a plug-in parameter will vary depending on the GUI framework you choose, there are a few basic design principles that should always be followed.

The basic DemoGain plug-in does not include any GUI implementation. For practical GUI implementation examples, open the DemoGain_GUIExtensions sample plug-ins. DemoGain_Cocoa provides an example of a custom plug-in GUI using the native macOS SDK, while DemoGain_Win32 uses native Windows APIs. The other examples in this folder require common third-party libraries.

12.2.5.5.1 Control edits vs. parameter updates The most important principle for AAX GUI design is that an edit in a plug-in's GUI should never directly set the state of the associated parameter or parameters. This is because there may be other controller clients of the plug-in's data model which will need to be notified of the edit, or which may need to override edits from the GUI.

- On control edit

When a control is edited in the plug-in GUI, call `AAX_IEffectParameters::SetParameterNormalizedValue()`. The implementation of this method should post a request to the host in order to trigger the parameter update. The GUI should not update its state until the corresponding parameter update notification is received.

- On parameter update

A parameter update can occur in response to a GUI edit, an edit from another attached controller, an update to a linked parameter, or any other event that affects the state of the data model. When the state of a parameter changes, `AAX_IEffectGUI::ParameterUpdated()` is called. The plug-in's GUI should be updated in this method in order to reflect the new state of the affected parameter.

For detailed information about the sequence of events and GUI responsibilities during a parameter update, see [Parameter updates](#)

12.2.5.5.2 Notifying the host of GUI events Some GUI events must be handled by the host rather than by the plug-in. For example, in Pro Tools a user should be able to display a pop-up menu for controlling automation information by command-option-control-clicking (Mac) or alt-ctl-right clicking (Windows) a control. These events, and other direct communication between the GUI and the "container" in which the host creates the plug-in view, is accomplished via the `AAX_IViewContainer` interface. Be sure to always call the handler methods in this interface before handling a mouse event within the plug-in GUI, in order to maintain the expected host behavior.

12.2.6 Next Steps

Now that you have a basic understanding of AAX, head back to the [front page](#) and continue reading through the suggested documentation. Or dig in to the [sample plug-ins](#) to see how specific features are supported in AAX.

Warning

Your AAX plug-ins will not be compatible with shipping versions of Pro Tools until they are digitally signed using tools provided by PACE Anti-Piracy, Inc. As an AAX developer you can receive these tools free of charge. Read the [Digital signature](#) section of the [Pro Tools Guide](#) to learn about the digital signing requirements for compatibility with Pro Tools.

Collaboration diagram for Getting Started with AAX:



12.3 Core AAX Interface

12.3.1

Main classes, callbacks, and format specification details for a standard AAX plug-in.

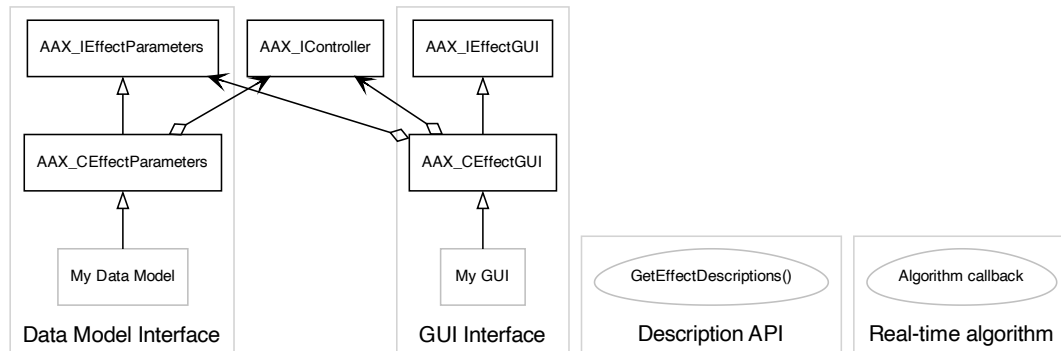


Figure 12.1 Main classes and callbacks for a standard AAX plug-in

These interfaces and components represent the standard interface between an AAX plug-in and the host. Default implementations for each interface are provided in the SDK. See the following modules for more information about each component.

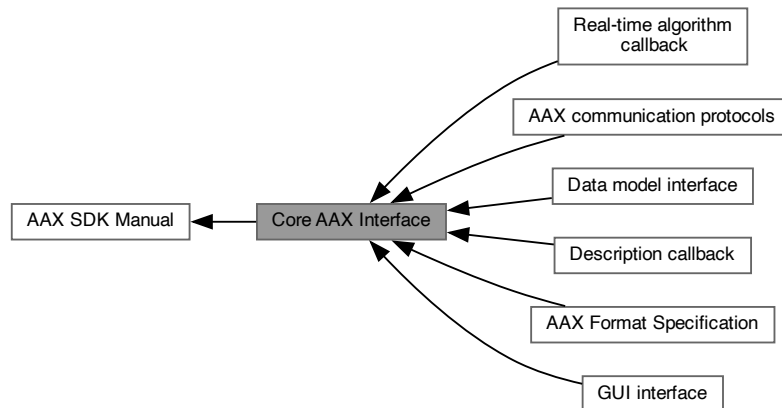
See also

[Component Structure](#) in the [Getting Started Guide](#)

Documents

- [Description callback](#)
Static configuration for an AAX plug-in.
- [Real-time algorithm callback](#)
A plug-in's audio processing core.
- [Data model interface](#)
- [GUI interface](#)
The interface for a AAX Plug-in's user interface.
- [AAX communication protocols](#)
How to transfer data between different parts of an AAX plug-in.
- [AAX Format Specification](#)
Additional requirements for AAX plug-ins.

Collaboration diagram for Core AAX Interface:



12.4 Description callback

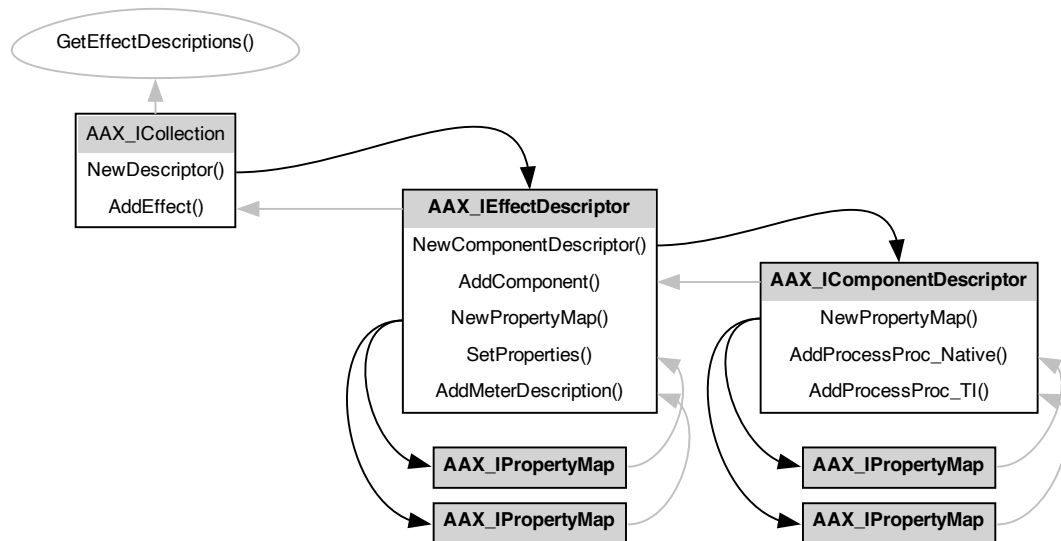
12.4.1

Static configuration for an AAX plug-in.

12.4.2 On this page

- [About the Describe callback and AAX descriptor interfaces](#)
- [Top level: Collection](#)
- [Middle level: Effects](#)
- [Lowest level: Algorithm components](#)
- [Checking Results](#)
- [Describe Validation](#)
- [Additional Topics](#)

12.4.3 About the Describe callback and AAX descriptor interfaces



In Describe, a plug-in declares its static (or default) configuration and any properties that the host will need in order to manage the plug-in.

A plug-in's Describe callback ties the Algorithm, GUI, and Data Model together. In this callback, the plug-in provides a description of its algorithm callbacks, data connections, and other static plug-in properties using a set of host-provided description interfaces. The plug-in uses a tiered hierarchy of these description interfaces to complete its description:

- At the top level, a single Collection interface represents properties that apply to the plug-in binary.
- The Collection interface is populated with one or more Effect descriptors, each of which represents a single kind of effect. For example, a single dynamics plug-in binary may include both single-band and multi-band Effects. Each Effect represents a different plug-in "product" available to users.
- Each Effect is registered with a set of one or more algorithm components that represent the specific processing configurations (e.g. stem formats, sample rates) that the plug-in supports. The set of components in a single Effect provide possible variations of the single plug-in "product", and as such these variations are mostly transparent to users.

The actual description callback entrypoint is [ACFRegisterPlugin\(\)](#), which is declared in [AAX_Exports.cpp](#). This method is implemented inside of the AAX Library, where it calls the plug-in's custom implementation of the [GetEffectDescriptions\(\)](#) callback.

```

// *****
// ROUTINE: GetEffectDescriptions
// *****
AAX_Result GetEffectDescriptions( AAX_ICollection * outCollection )
{
    AAX_Result result = AAX_SUCCESS;
    AAX_IEffectDescriptor * plugInDescriptor = outCollection->NewDescriptor();
    if ( plugInDescriptor )
    {
        result = DemoGain_GetPlugInDescription( plugInDescriptor );
        if ( result == AAX_SUCCESS )
            outCollection->AddEffect( kEffectID_DemoGain, plugInDescriptor );
    }
}

```

```

else result = AAX_ERROR_NULL_OBJECT;

outCollection->SetManufacturerName( "Avid, Inc." );
outCollection->AddPackageName( "DemoGain Plug-In" );
outCollection->AddPackageName( "DemoGain" );
outCollection->AddPackageName( "DmGi" );
outCollection->SetPackageVersion( 1 );

return result;
}

```

[GetEffectDescriptions\(\)](#) from the DemoGain example plug-in

In general, the following procedure is used when describing an AAX plug-in:

1. Create an Effect description interface from the Collection interface provided by the host
2. Use the Effect description interface to create references to one or more component description interfaces.
3. Describe each algorithm component by populating the component descriptions.
4. Add the components to the Effect description
5. Add additional modules and properties to the Effect description.
6. Add the completed Effect to the Collection
7. Repeat for any additional Effects included in the plug-in binary.
8. Return the completed Collection interface to the host and exit.

Note

The host owns all memory associated with any descriptors that the plug-in returns via this callback.

12.4.4 Top level: Collection

The [AAX_ICollection](#) interface provides a creation function ([AAX_ICollection::NewDescriptor\(\)](#)) for new plug-in descriptors, which in turn provides access to the various interfaces necessary for describing a plug-in (see the [DemoGain_GetPlugInDescription\(\)](#) and [DescribeAlgorithmComponent\(\)](#) listings below).

When a plug-in description is complete, it is added to the collection via the [AAX_ICollection::AddEffect\(\)](#) method. The [AAX_ICollection](#) interface also provides some additional description methods that are used to describe the overall plug-in package. These methods can be used to describe the plug-in package's name, the name of the plug-in's manufacturer, and the plug-in package version. Once these have been described, the completed description interface is returned to the host and exits.

```

// *****
// ROUTINE: GetEffectDescriptions
// *****
AAX_Result GetEffectDescriptions( AAX_ICollection * outCollection )
{
    .
    .
    .
    AAX_IEffectDescriptor *   plugInDescriptor = outCollection->NewDescriptor();
    .
    .
    .
    result = DemoGain_GetPlugInDescription( plugInDescriptor );
    .
    .
    .
    outCollection->AddEffect( kEffectID_DemoGain, plugInDescriptor );
    .
    .
    .
    outCollection->SetManufacturerName( "Avid, Inc." );
}

```

```

outCollection->AddPackageName( "DemoGain Plug-In" );
outCollection->AddPackageName( "DemoGain" );
outCollection->AddPackageName( "DmGi" );
outCollection->SetPackageVersion( 1 );
    .
    .
    .
return result;
}

```

Populating the [AAX_ICollection](#) interface

12.4.5 Middle level: Effects

The [AAX_IEffectDescriptor](#) interface provides description methods that are used to describe the Effect, such as its name, category, associated page table, and, importantly, creation methods for its data model, GUI, and other AAX modules.

```

// *****
// ROUTINE: DemoGain_GetPlugInDescription
// *****
static AAX_Result DemoGain_GetPlugInDescription( AAX_IEffectDescriptor * outDescriptor )
{
    int err;
    AAX_IComponentDescriptor * compDesc = outDescriptor->NewComponentDescriptor ();
    if ( !compDesc )
        return AAX_ERROR_NULL_OBJECT;

    // Add empty component descriptors to the host, register a processing
    // entrypoint for each, and populate with description information.
    //
    // Alg component
    DescribeAlgorithmComponent( compDesc );
    err = outDescriptor->AddComponent( compDesc ); AAX_ASSERT (err == 0);

    outDescriptor->AddPlugInName ( "Demo Gain AAX" );
    outDescriptor->AddPlugInName ( "Demo Gain" );
    outDescriptor->AddPlugInName ( "DemoGain" );
    outDescriptor->AddPlugInName ( "DmGain" );
    outDescriptor->AddPlugInName ( "DGpr" );
    outDescriptor->AddPlugInName ( "Dn" );
    outDescriptor->AddPlugInCategory ( AAX_ePlugInCategory_Dynamics );
    outDescriptor->AddProcPtr( (void *) DemoGain_Parameters::Create, kAAX_ProcPtrID_Create_EffectParameters
    );
    outDescriptor->AddResourceInfo ( AAX_eResourceType_PageTable, "DemoGainPages.xml" );

#ifdef PLUGGUI != 0
    outDescriptor->AddProcPtr( (void *) DemoGain_GUI::Create, kAAX_ProcPtrID_Create_EffectGUI );
#endif

    return AAX_SUCCESS;
}

```

Populating an [AAX_IEffectDescriptor](#) interface

All components in an Effect must share the same AAX modules; for example, it is not possible to use one data model definition for one sample rate and another data model definition for a different sample rate. Therefore, a plug-in's AAX modules are defined in its Effect description.

12.4.5.1 Registering multiple Effects

A single plug-in package may include multiple Effects, which are added in turn in the description method. Once these connections are made, Describe passes the host a populated description interface and returns.

For example, consider an EQ plug-in that contains both one-band and four-band variations, each of which the user should see as a distinct plug-in. These Effects would be described and added separately to the collection object and would appear as separate products to the user.

```

AAX_Result GetEffectsDescriptions ( AAX_ICollection * outCollection )
{
    AAX_Result result = AAX_SUCCESS ;
    if ( result == AAX_SUCCESS )

```

```

{
    AAX_IEffectDescriptor * aDesc1 = outCollection -> NewDescriptor ();
    // ...
    // Populate aDesc1 with one - band EQ description
    // ...
    result = outCollection -> AddEffect ( kEffectID_MyOneBandEQ , aDesc1 );
}
if ( result == AAX_SUCCESS )
{
    AAX_IEffectDescriptor * aDesc4 = outCollection -> NewDescriptor ();
    // ...
    // Populate aDesc4 with four - band EQ description
    // ...
    result = outCollection -> AddEffect ( kEffectID_MyFourBandEQ , aDesc4 );
}
if ( result == AAX_SUCCESS )
{
    outCollection -> SetManufacturerName ( "My Plug -Ins , Inc." );
    outCollection -> AddPackageName ( " MyEQ Plug -In" );
    outCollection -> AddPackageName ( " MyQ" ); // Short name
    outCollection -> SetPackageVersion ( 1 );
}
return result ;
}

```

Registering multiple Effects in a single Collection

12.4.6 Lowest level: Algorithm components

In order to register an algorithm component, a plug-in must describe the component's external interface. This includes each of the component's ports and any other fields in its context structure, a reference to its processing function entrypoint (its "Process Procedure", or ProcessProc,) and any other special properties that the host should know about.

The description of a context structure involves a set of port definitions, which can be "hard-wired" to receive data from the host (such as audio buffers), from the plug-in's data model (such as packets of coefficients), or even from past calls to the algorithm (private, persistent algorithm state). See [Real-time algorithm callback](#) for more information on an algorithm's context structure.

```

static void DescribeAlgorithmComponent( AAX_IComponentDescriptor * outDesc )
{
    .
    .
    .
    AAX_Result err;

    // Subscribe context fields to host-provided services or information
    err = outDesc->AddField ( eAlgFieldID_AudioIn, kAAX_FieldTypeAudioIn ); AAX_ASSERT(err == 0);
    err = outDesc->AddField ( eAlgFieldID_AudioOut, kAAX_FieldTypeAudioOut ); AAX_ASSERT (err == 0);
    err = outDesc->AddField ( eAlgFieldID_BufferSize, kAAX_FieldTypeAudioBufferLength ); AAX_ASSERT (err == 0);

    // Register context fields as communications destinations
    err = outDesc->AddDataInPort ( eAlgPortID_BypassIn, sizeof (int32_t) ); AAX_ASSERT (err == 0);
    err = outDesc->AddDataInPort ( eAlgPortID_CoefsGainIn, sizeof (SDemoGain_CoefsGain) ); AAX_ASSERT (err == 0);
    .
    .
    .
}

```

Populating a single [AAX_IComponentDescriptor](#) interface

12.4.6.1 Algorithm callback properties

A set of callback properties is required when adding a Process Procedure to an algorithm component. This is done via the [AAX_IPropertyMap](#) interface. Using distinct property maps, a single component may register multiple versions of its callback. For example, an audio processing component might register mono and stereo callbacks, or Native and TI callbacks, assigning each the applicable property mapping. This allows the host to determine the correct callback to use depending on the environment in which the plug-in is instantiated.

```

static void DescribeAlgorithmComponent( AAX_IComponentDescriptor * outDesc )

```

```

{
    AAX_IPropertyMap *      properties = outDesc->NewPropertyMap();
    .
    .
    .
    properties->Clear ();
    properties->AddProperty ( AAX_eProperty_ManufacturerID, cDemoGain_ManufactureID );
    properties->AddProperty ( AAX_eProperty_ProductID, cDemoGain_ProductID );
    properties->AddProperty ( AAX_eProperty_InputStemFormat, AAX_eStemFormat_Mono );
    properties->AddProperty ( AAX_eProperty_OutputStemFormat, AAX_eStemFormat_Mono );
    properties->AddProperty ( AAX_eProperty_CanBypass, true );

    // Native and AudioSuite versions
    properties->AddProperty ( AAX_eProperty_PluginID_Native, cDemoGain_PluginID_Native );
    properties->AddProperty ( AAX_eProperty_PluginID_AudioSuite, cDemoGain_PluginID_AudioSuite ); // Since
    this is a linear plug-in the RTAS version can also be an AudioSuite version.
    properties->AddProperty ( AAX_eProperty_DSP_AudioBufferLength, kAAX_NativeAudioBufferLength_Default );
    err = outDesc->AddProcessProc_Native ( DemoGain_AlgorithmProcessFunction <1, 1,
    1<kAAX_NativeAudioBufferLength_Default>, properties ); AAX_ASSERT (err == 0);

    // TI DSP Version
    properties->AddProperty ( AAX_eProperty_PluginID_TI, cDemoGain_PluginID_TI );
    properties->AddProperty ( AAX_eProperty_DSP_AudioBufferLength, AAX_eAudioBufferLengthDSP_Default );
    properties->AddProperty ( AAX_eProperty_TI_InstanceCycleCount, 1055 );
    properties->AddProperty ( AAX_eProperty_TI_SharedCycleCount, 58 );
}

```

Adding properties to a component description

AAX does not require that every value in [AAX_IPropertyMap](#) be assigned by the developer. However, if a specific value is not assigned to one of an element's properties then the element must support any value for that property. For example, if an audio processing component does not provide any stem format properties then the host will assume that the callback will support any stem format.

12.4.7 Checking Results

12.4.7.1 Summary

- Use [AAX_CheckedResult](#) to store result values from all method calls in Describe.
- Use the [AAX_SWALLOW](#) and [AAX_SWALLOW_MULT](#) macros to encapsulate independent describe code, such as registration logic for separate Effects or for separate ProcessProc variations within a single Effect.

12.4.7.2 The Problem

With plain [AAX_Result](#) values it can be challenging to properly detect and handle error states. Each description method call returns an [AAX_Result](#) to indicate success or failure, and often problems in a plug-in's configuration can be addressed by properly detecting and resolving errors that occur here. However, adding a return value check after every method and providing conditional logic in the case of a failure is onerous, ugly, and difficult to maintain.

```
AAX_Result result = AAX_SUCCESS;
```

```

result = descriptor->SomeMethod();
result = descriptor->AnotherMethod(); // oops! might ignore an error
// -----
// Safer, but not a good solution:
// Information about the errors is lost, the
// merged error code is meaningless, and
// debugging to find the location of the
// failure is hard.
//
result |= descriptor->SomeMethod();
result |= descriptor->AnotherMethod();
// ...
if (AAX_SUCCESS != result)
{
    // handle the merged error code
}
// -----
// This is also not a good solution:
// There is no actual handling of errors

```

```
// (from the SDK example plug-ins)
//
result = descriptor->SomeMethod();
AAX_ASSERT(AAX_SUCCESS == result);
result = descriptor->AnotherMethod();
AAX_ASSERT(AAX_SUCCESS == result);
// -----
// This is correct but is too hard
//
AAX_Result result = AAX_SUCCESS;
result = descriptor->SomeMethod();
if (AAX_SUCCESS != result) {
    // logic to handle this error:
    // assert and/or log the failure?
    // return or continue execution?
}

result = descriptor->AnotherMethod();
if (AAX_SUCCESS != result) {
    // ditto
}
```

[AAX_Result](#) based error checking is awkward

12.4.7.3 The Solution

The [AAX_CheckedResult](#) class is designed to solve this problem. [AAX_CheckedResult](#) can be used just like a plain-old-data [AAX_Result](#) :

```
AAX_CheckedResult result = AAX_SUCCESS;
result = descriptor->SomeMethod();
result = descriptor->AnotherMethod();
```

Simpler result checking with [AAX_CheckedResult](#)

When a failure is encountered, [AAX_CheckedResult](#) will:

- Store the error value
- Log the error using `AAX_TRACE_RELEASE`
- Throw an exception of type [AAX_CheckedResult::Exception](#)

To make this safe to use in the Describe routine, the [AAX](#) Library includes a try/catch block around the call to the plug-in's `GetEffectDescriptions()` routine.

Warning

Do not use [AAX_CheckedResult](#) anywhere where an exception could escape to the host (`GetEffectDescriptions()` is OK)

12.4.7.4 Handling Errors and Managing Control Flow

With the basic approach shown above, any error which is encountered will throw an exception which will cancel the plug-in's registration and prevent the plug-in from being shown in the host. However, most errors can be safely handled without canceling the entire plug-in registration.

```
AAX_CheckedResult result = AAX_SUCCESS;

// effect 1 registration
result = DescribeMyEffect1( effect1Descriptor );
result = outCollection->AddEffect( myEffect1ID, effect1Descriptor );

// effect 2 registration
result = DescribeMyEffect2( effect2Descriptor );
result = outCollection->AddEffect( myEffect2ID, effect2Descriptor );
```


Example with no error handling

In this example, a failure when describing either individual effect will prevent the other effect from being registered. Registration of individual ProcessProcs within a single effect, e.g. for different stem formats, is similar.

To allow the registration of other effects to proceed in the event of a failure, any exceptions thrown during the registration of one effect should be caught and should only prevent the registration of that individual effect.

```
AAX_CheckedResult result = AAX_SUCCESS;

// effect 1 registration
try {
    result = DescribeMyEffect1( effect1Descriptor );
    result = outCollection->AddEffect( myEffect1ID, effect1Descriptor );
}
catch (const AAX_CheckedResult::Exception& ex) {
    // log the error using ex.What()
    // swallow the exception and proceed
}

// effect 2 registration
try {
    result = DescribeMyEffect2( effect2Descriptor );
    result = outCollection->AddEffect( myEffect2ID, effect2Descriptor );
}
catch (const AAX_CheckedResult::Exception& ex) {
    // ditto
}
```

Example of error handling with try/catch

This solves the problem fully, but it is still cumbersome - especially when registering a long list of separate ProcessProc variants!

The [AAX_SWALLOW_MULT](#) macro makes it easier to handle errors which are thrown by [AAX_CheckedResult](#) :

```
AAX_CheckedResult result = AAX_SUCCESS;

// effect 1 registration
AAX_SWALLOW_MULT (
    result = DescribeMyEffect1( effect1Descriptor );
    result = outCollection->AddEffect( myEffect1ID, effect1Descriptor );
);

// effect 2 registration
AAX_SWALLOW_MULT (
    result = DescribeMyEffect2( effect2Descriptor );
    result = outCollection->AddEffect( myEffect2ID, effect2Descriptor );
);
```

Example of error handling with [AAX_SWALLOW_MULT](#)

Variations

- For single-line try/catch there is also [AAX_SWALLOW](#).
- If you need to reference the error value after the exception is caught, use [AAX_CAPTURE_MULT](#) (multi-line) or [AAX_CAPTURE](#) (single-line)
- If you know that a certain error code is OK and should not throw in a given situation then you can add it as an exception to the [AAX_CheckedResult](#) object with [AAX_CheckedResult::AddAcceptedResult\(\)](#).

For examples of [AAX_CheckedResult](#) in use, see the [DemoGain_Multichannel](#) and [DemoGain_UpMixer](#) plug-ins

12.4.8 Describe Validation

12.4.8.1 Validation with DSH

You can validate your plug-in's Describe routine using the [DigiShell](#) command-line tool. The validation command is available directly in the aaxh dish and is also available through an AAX Validator test module:

aaxh dish

```
dsh> load_dish aaxh
dsh> loadpi "/quoted/path/without escape chars/MyPlugIn.aaxplugin"
dsh> getdescriptionvalidationinfo 0
```

AAX Validator

```
dsh> load_dish aaxval
dsh> runtest [test.describe_validation, "/quoted/path/without escape chars/MyPlugIn.aaxplugin"]
```

12.4.8.2 Validation with Pro Tools

Beginning in Pro Tools 12.8.2, developer builds of Pro Tools will also check plug-in describe routines and will present an alert dialog when the plug-in is scanned if any aspect of the plug-in's describe code has failed the validation step.

Describe validation warning in a Pro Tools developer build

The specific kinds of errors which were encountered will be printed to the [DigiTrace](#) log file:

```
13033.502646,00307,0073: ERROR: Unknown target host for the plug-in.
13033.502662,00307,0073: ERROR: PlugInID property is missing for a ProcessProc (process, initialization, or ba
13033.502734,00307,0e0f: CMN_TRACEASSERT Sandbox.aaxplugin configuration contains 2 errors. See the DigiTrace
```

This check may be suppressed using the following [DigiOption](#):

```
TestPlugInDescriptions 0
```

12.4.9 Additional Topics

See also

[Plug-in meters](#)

Classes

- class [AAX_ICollection](#)
Interface to represent a plug-in binary's static description.
- class [AAX_IComponentDescriptor](#)
Description interface for an AAX plug-in component.
- class [AAX_IEffectDescriptor](#)
Description interface for an effect's (plug-in type's) components.
- class [AAX_IPropertyMap](#)
Generic plug-in description property map.

Functions

- [AAX_Result AAXRegisterPlugin](#) ([IACFUnknown](#) *pUnkHost, [IACFPluginDefinition](#) **ppPluginDefinition)
The main plug-in registration method.
- [AAX_Result GetEffectDescriptions](#) ([AAX_ICollection](#) *inCollection)
The plug-in's static Description endpoint.

12.4.10 Function Documentation

12.4.10.1 AAXRegisterPlugin()

```
AAX_Result AAXRegisterPlugin (
    IACFUnknown * pUnkHost,
    IACFPluginDefinition ** ppPluginDefinition )
```

The main plug-in registration method.

This method determines the number of components defined in the dll. The implementation of this method in the AAX library calls the following function, which must be implemented somewhere in your plug-in:

```
extern AAX_Result GetEffectDescriptions( AAX_ICollection * outCollection );
```

Wrapped by [ACFRegisterPlugin\(\)](#)

Referenced by [ACFRegisterPlugin\(\)](#).

Here is the caller graph for this function:



12.4.10.2 GetEffectDescriptions()

```
AAX_Result GetEffectDescriptions (
    AAX_ICollection * inCollection )
```

The plug-in's static Description endpoint.

This function is responsible for describing an AAX plug-in to the host. It does this by populating an [AAX_ICollection](#) interface.

This function must be included in every plug-in that links to the AAX library. It is called when the host first loads the plug-in.

Parameters

out	inCollection	
-----	--------------	--

Collaboration diagram for Description callback:



12.5 Real-time algorithm callback

A plug-in's audio processing core.

12.5.1 On this page

- [Algorithm definition](#)
- [Algorithm memory management](#)
- [Communicating with the algorithm](#)
- [Algorithm initialization](#)
- [Algorithm processing](#)
- [Persistent algorithm memory](#)
- [Example algorithm callback](#)
- [Port Types and Behavior](#)
- [Additional Information](#)

12.5.2 Algorithm definition

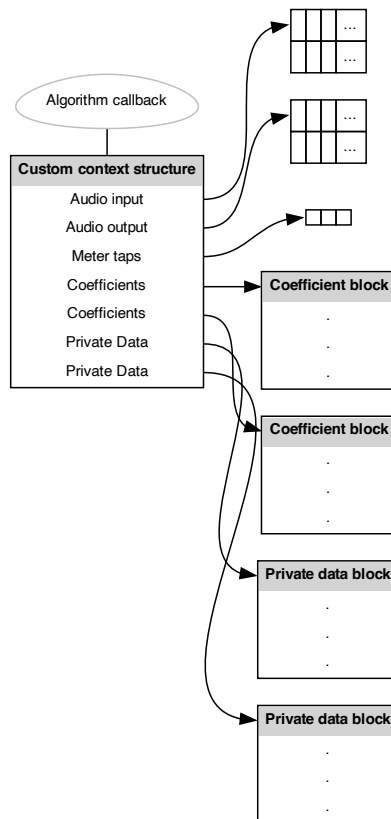
Algorithm processing in AAX plug-ins is handled via a C-style algorithm processing callback (see code below.) Each Effect variation in a plug-in must register an algorithm entrypoint in the plug-in [description](#), and the host will call this entrypoint to render a buffer of audio samples.

```

void AAX_CALLBACK MyPlugIn_AlgorithmProcessFunction(
    SMyPlugIn_Alg_Context * const inInstancesBegin [],
    const void * inInstancesEnd)
{
    //
    // Processing code...
    //
}
  
```

12.5.3 Algorithm memory management

This callback pattern is designed such that plug-in algorithms may be easily loaded in remote memory spaces on a variety of devices and quickly re-compiled for different operating environments without significant changes to the code, and this design goal informs the algorithm's memory management techniques.



When the AAX host calls a plug-in's algorithm callback, it provides a block of memory describing the state of the plug-in. This block of memory, known as the algorithm's *context*, can be thought of as the algorithm's interface to the outside world: when another part of the plug-in interacts with the algorithm, it does so by posting information to the algorithm's context.

```
//=====
// Component context definitions
//=====

// Context structure
struct SDemoDistAlg_Context
{
    int32_t          * mCtrlBypassP;           // Coefficient message destination
    float            * mCtrlMixP;              // Coefficient message destination
    float            * mCtrlInpGP;             // Coefficient message destination
    float            * mCtrlOutGP;             // Coefficient message destination
    SDemoDist_DistCoefs * mCoefsDistP;          // Coefficient message destination
    SDemoDist_FiltCoefs * mCoefsFiltP;          // Coefficient message destination

    CSimpleBiquad     * mBiquads;               // Private data

    float*            * mInputPP;               // Audio signal input
    float*            * mOutputPP;              // Audio signal output
    float*            * mMeterTapsPP;           // Meter signal output
};
```

It is important to note that, in most circumstances, algorithm callbacks do not own their own memory. The algorithm and its memory is managed entirely by the host or shell environment, and relies on the host-provided context structure for all state information.

If persistent memory is required, algorithms can register for block(s) of persistent state data via the [AAX_IComponentDescriptor::AddPrivateData\(\)](#) API (as in `SDemoDist_Alg_Context::mBiquads` above.) A plug-in may store state data in the resulting "private data" context fields and this data will be restored by the host when the algorithm is next called. See the [Persistent algorithm memory](#) section below for more information.

12.5.4 Communicating with the algorithm

Plug-ins communicate with their algorithms via a buffered, host-managed message system. The host guarantees that messages posted to this system will be delivered to the applicable context field and that the algorithm's context is up to date every time the component is entered.

This system utilizes a static data routing scheme that is defined in the plug-in's `describe` method. Once the routing scheme has been defined, the plug-in may post packets of data to its algorithm using [AAX_IController::PostPacket\(\)](#).

In order to reference the fields in its algorithm's context, the plug-in's host-side code uses unique identifiers generated with the [AAX_FIELD_INDEX](#) macro:

```
enum EDemoDist_Alg_PortID
{
    eAlgPortID_BypassIn           = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mCtrlBypassP)
    ,eAlgPortID_MixIn             = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mCtrlMixP)
    ,eAlgPortID_InpGIn            = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mCtrlInpGP)
    ,eAlgPortID_OutGIn            = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mCtrlOutGP)
    ,eAlgPortID_CoefsDistIn       = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mCoefsDistP)
    ,eAlgPortID_CoefsFilterIn     = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mCoefsFiltP)

    ,eAlgFieldID_Biquads          = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mBiquads)

    ,eDemoDist_AlgFieldID_AudioIn = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mInputPP)
    ,eDemoDist_AlgFieldID_AudioOut = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mOutputPP)
    ,eAlgFieldID_MeterTaps        = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mMeterTapsPP)
};
```

See [Description callback](#) for more information about registering context fields and defining a plug-in's message routing scheme.

12.5.5 Algorithm initialization

The following events occur before the AAX host begins calling a plug-in's algorithm:

- The Effect's [data model](#) is initialized
- An initial call to [AAX_IEffectParameters::ResetFieldData\(\)](#) is made for each private data block in the algorithm.
- An initial call to [AAX_IEffectParameters::GenerateCoefficients\(\)](#) is made and coefficient packets are dispatched to each of the algorithm's data ports based on the default model state.
- All packets are delivered and initial algorithm context state is set
- If one has been registered, the algorithm's optional initialization callback is called with the default context
- (Algorithmic processing begins)

12.5.5.1 Private data initialization

To initialize an algorithm's private data blocks, [AAX_IEffectParameters::ResetFieldData\(\)](#) is called on the host for each block in the algorithm. The host uses this method to acquire a default initialized memory block for each private data port, which is then copied into the algorithm's memory pool and provided to its context.

The default implementation of this method in [AAX_CEffectParameters](#) will initialize the data to zero.

See also

[Persistent algorithm memory](#)

12.5.5.2 Optional initialization callback

If any additional initialization or de-initialization steps are required for proper operation of the algorithm, an optional initialization routine may be registered and associated with the algorithm's processing callback. This initialization routine will be called in the same device / memory space as the algorithm's processing context. The initialization callback is provided with the algorithm's default context and is called both before every new instance of the Effect begins its algorithm render callbacks and before every instance is destroyed.

This initialization routine is provided in [Describe](#) as an argument to the platform's `AddProcessProc` registration method:

- [AAX_IComponentDescriptor::AddProcessProc_Native\(\)](#)
- [AAX_IComponentDescriptor::AddProcessProc_Tl\(\)](#)

Host Compatibility Notes As of Pro Tools 10.2.1 an algorithm's initialization callback routine will have up to 5 seconds to execute.

See also

[AAX_CInstanceInitProc](#)

12.5.6 Algorithm processing

Once the algorithm has been initialized and processing begins, the algorithm function is called regularly by the host audio engine. The algorithm may read the context data provided by the host and is responsible for writing data to all of the samples in its output buffers each time it is executed.

Note

The data in an algorithm's output buffers is not initialized before the algorithm is called, thus the algorithm must always write data into all output samples. This is to ensure equivalent behavior between all platforms, some of which do not have the resource budget to pre-initialize output data buffers.

12.5.7 Persistent algorithm memory

An AAX plug-in algorithm may contain one or more *private data* ports in its context. These are the only context fields in which an algorithm may store persistent state data.

12.5.7.1 Private memory characteristics

Each private data port is a pointer to a preallocated block of memory. The size of each port is defined during [Describe](#) when the port is registered. On DSP systems, the plug-in may request that the data block be placed in the chip's external memory.

Once private data is allocated by the plug-in host or DSP shell, it will not be relocated or re-allocated until the algorithm is destroyed (see [Optional initialization callback](#))

12.5.7.2 Private data port registration

Private data ports are registered during Describe via [AAX_IComponentDescriptor::AddPrivateData\(\)](#). This method defines the size of the data block that will be allocated as well as an initialization callback with format [AAX_CInitPrivateDataProc](#).

12.5.7.3 Private data initialization

[AAX_IEffectParameters::ResetFieldData\(\)](#) is called on the host for both Native and DSP plug-ins. For DSP plug-ins, the initialized data block is copied to the DSP by the AAX host following the initialization callback. The initialization callbacks for a plug-in's private data blocks are called after all host modules have been initialized and before the algorithm's optional initialization callback.

See also

[Algorithm initialization](#)

12.5.7.4 Private data communication

It is possible to transfer data to and from the algorithm's private data blocks using the [AAX_IPrivateDataAccess](#) interface, which is available in a TimerWakeup context through the [AAX_IEffectDirectData](#) interface. For more information about this API, see [auxinterface_directdata_privatedataaccess](#).

12.5.8 Example algorithm callback

As a final example, the code below describes a simple audio processing component. The component's context contains one message pointer to receive incoming "gain" parameter values, as well as one audio data input, "pdI", and one audio data output, "pdO". Additionally there is a message pointer to receive "bypass" on/off values. The host calls the component each time a new input sample buffer must be processed, and each time the component is called the host ensures that all context fields are up-to-date.

```
void AAX_CALLBACK MyPlugIn_AlgorithmProcessFunction(
    SMyPlugIn_Alg_Context * const    inInstancesBegin [],
    const void *                    inInstancesEnd)
{
    // Get a pointer to the beginning of the memory block table
    SMyPlugIn_Alg_Context* AAX_RESTRICT instance = inInstancesBegin [0];

    //----- Iterate over plug-in instances -----//
    for (SMyPlugIn_Alg_Context * const * walk = inInstancesBegin; walk < inInstancesEnd; ++walk)
    {
        instance = *walk;

        //----- Retrieve instance-specific information -----//
        //
        const SMyPlugIn_CoefsGain* const AAX_RESTRICT  coefsGainP =    instance->mCoefsGainP; // Input
    (const)
        const int32_t      bypass      = *instance->mCtrlBypassP;
        const float        gain        = coefsGainP->mGain;

        //----- Run processing loop over each input channel -----//
        //
        for (int ch = 0; ch < kNumChannelsIn; ch++) // Iterate over all input channels
        {
            //----- Run processing loop over each sample -----//
            //
            for (int t = 0; t < kAudioWindowSize; t++)
            {
                float* const AAX_RESTRICT pdI = instance->mInputPP [ch];
                float* const AAX_RESTRICT pdO = instance->mOutputPP [ch];

                if ( pdI && pdO )
                {
                    pdO [t] = gain * pdI [t];
                    if (bypass) { pdO [t] = pdI [t]; }
                }
            } // Go to the next sample
        } // Go to next channel
    } // End instance-iteration loop
}
```


12.5.9 Port Types and Behavior

In this section, we will examine the various kinds of ports that can be used by the algorithm component in an AAX plug-in:

1. Standard message input
2. Internal state storage
3. Metering output
4. Environment variable retrieval
5. Other functionality enhancement

12.5.9.1 Standard message input

Most ports will function as pointers to incoming data. This data can have any type. For example, an algorithm's context may include a port of type `float*` to receive incoming float data and another port of type `SMyCustomStructure*` to receive incoming `SMyCustomStructure` data.

Like all registered context fields, input ports are managed by the hosting environment such that they always point to the most recently received data at the time that the algorithm callback is entered. The algorithm may not store or alter data in a standard message input port: this data is available as read-only input. If data is stored in the space allocated for the port's data then the result will be undefined behavior.

To define a standard message input port, a plug-in should call `AAX_IComponentDescriptor::AddDataInPort()`.

12.5.9.2 Internal state storage

Most plug-ins require local data to be accessible to their algorithms. These may be static data, such as lookup tables, or dynamic data, such as coefficient smoothing history or delay lines. In the `DemoDist` sample plug-in, `SDemoDist_Alg_Context::mBiquads` is an example of this type of port: it is not modified by any other component and `DemoDist_AlgorithmProcessFunction()` relies on the `mBiquads` data persisting between processing calls.

A component that has registered a private data field is given access to a block of private data. Although the memory in this block will be allocated by the host, its data is fully owned by the component. Because this data is considered private to its parent component, other components cannot overwrite or target this data. Plug-ins that need to transmit data directly between their algorithms' private data ports and their other modules may use the `AAX_IEffectDirectData` interface, which provides an API for reading from or writing to this data from outside of the algorithm callback.

The plug-in's data model includes an initialization function that is called by the AAX host at the time of plug-in instantiation and "reset" events. This initialization method is called on the host for both Native and DSP plug-ins. Since this method is part of the plug-in's data model, it has direct access to plug-in state information.

12.5.9.3 Metering output

Plug-in metering ports are populated with an array of float values, or 'taps'. One tap is provided per plug-in meter. The algorithm writes per-buffer peak values to this port and the AAX host applies standardized ballistics to these values. Both raw and processed meter values are available to the plug-in's GUI.

12.5.9.4 Environment variable retrieval

Another use of ports is to receive data from the AAX host describing the execution environment. For example, an algorithm may include a port to receive the number of samples in its processing window or the sample rate. These services are provided automatically by the host once the component registers ports for them.

12.5.9.5 Other functionality enhancement

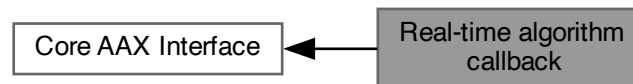
An algorithm component may use ports to gain additional functionality that is provided by the host. For example, an algorithm that will be compiled for accelerated environments may take advantage of the TI chip's Direct Memory Access functionality by registering a DMA port. The host will then allow this port to access memory directly using [AAX's DMA APIs](#).

12.5.10 Additional Information

For information about optional features for the algorithm processing callback, see the following [Additional AAX features](#) documentation:

- [Direct Memory Access](#)
- [Background processing callback](#)

Collaboration diagram for Real-time algorithm callback:



12.6 Data model interface

12.6.1

The interface for an AAX Plug-in's data model.

[:Implemented by the Plug-In](#)

The interface for an instance of a plug-in's data model. A plug-in's implementation of this interface is responsible for creating the plug-in's set of parameters and for defining how the plug-in will respond when these parameters are changed via control updates or preset loads. In order for information to be routed from the plug-in's data model to its algorithm, the parameters that are created here must be registered with the host in the plug-in's [Description callback](#).

At [initialization](#), the host provides this interface with a reference to [AAX_IController](#), which provides access from the data model back to the host. This reference provides a means of querying information from the host such as stem format or sample rate, and is also responsible for communication between the data model and the plug-in's (decoupled) algorithm. See [Real-time algorithm callback](#).

You will most likely inherit your implementation of this interface from [AAX_CEffectParameters](#), a default implementation that provides basic data model functionality such as adding custom parameters, setting control values, restoring state, generating coefficients, etc., which you can override and customize as needed.

The following tags appear in the descriptions for methods of this class and its derived classes:

- **CALL**: Components in the plug-in should call this method to get / set data in the data model.

Note

- This class always inherits from the latest version of the interface and thus requires any subclass to implement all the methods in the latest version of the interface. The current version of [AAX_CEffectParameters](#) provides a convenient default implementation for all methods in the latest interface.
- Except where noted otherwise, the parameter values referenced by the methods in this interface are normalized values. See [Parameter Manager](#) for more information.

Legacy Porting Notes In the legacy plug-in SDK, these methods were found in `CProcess` and `CEffectProcess`. For additional `CProcess` methods, see [AAX_IEffectGUI](#).

12.6.2 Related classes

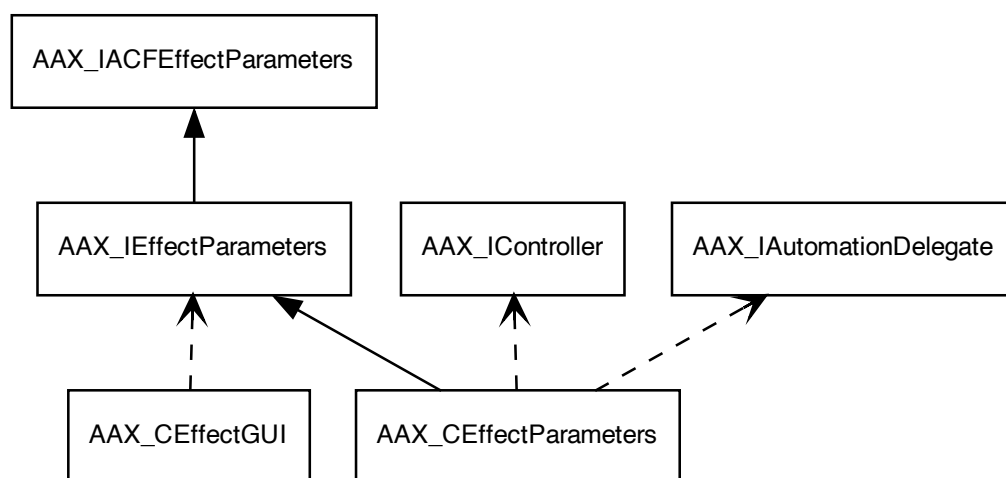


Figure 12.2 Classes related to **AAX_IEffectParameters** by inheritance or composition

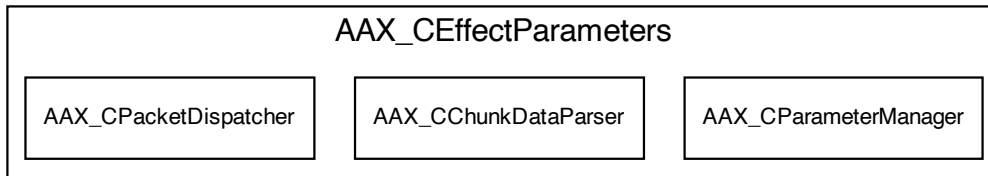


Figure 12.3 Classes owned as member objects of `AAX_CEffectParameters`

Classes

- class [AAX_CEffectParameters](#)
Default implementation of the [AAX_IEffectParameters](#) interface.
- class [AAX_IACFEffectParameters](#)
The interface for an AAX Plug-in's data model.
- class [AAX_IACFEffectParameters_V2](#)
Supplemental interface for an AAX Plug-in's data model.
- class [AAX_IACFEffectParameters_V3](#)
Supplemental interface for an AAX Plug-in's data model.
- class [AAX_IACFEffectParameters_V4](#)
Supplemental interface for an AAX Plug-in's data model.
- class [AAX_IEffectParameters](#)
The interface for an AAX Plug-in's data model.

Collaboration diagram for Data model interface:



12.7 GUI interface

12.7.1

The interface for a AAX Plug-in's user interface.

The [GUI interface](#) includes methods for handling the plug-in's GUI window and events.

Accessing the window

In AAX, the plug-in's window is provided as a native window pointer through the [AAX_IViewContainer](#) interface. The plug-in may also use this interface to forward events in its window back to the host for handling.

Default implementation

A default implementation of the GUI interface, [AAX_CEffectGUI](#), is compiled in to the AAX library. This class includes a few helper methods and other extensions to the base interface. Of particular note are several additional pure virtual methods that are used by this class to extend the GUI API, and which must be overridden by any inheriting class.

Extensions

The AAX SDK includes several examples of how the basic GUI interface may be extended to support native or third-party GUI frameworks. These examples are not a core part of the SDK, but are provided to developers as a convenience when incorporating their own chosen GUI framework.

Classes

- class [AAX_CEffectGUI](#)
Default implementation of the [AAX_IEffectGUI](#) interface.
- class [AAX_IACFEffectGUI](#)
The interface for a AAX Plug-in's GUI (graphical user interface).
- class [AAX_IEffectGUI](#)
The interface for a AAX Plug-in's user interface.
- class [AAX_IViewContainer](#)
Interface for the AAX host's view of a single instance of an effect. Used both by clients of the AAX host and by effect components.

Collaboration diagram for GUI interface:



12.8 AAX communication protocols

How to transfer data between different parts of an AAX plug-in.

AAX is a highly modular architecture. This section describes the various means by which AAX plug-in modules may communicate with one another and with the host.

There are two fundamental categories of communication in [AAX](#):

1. [Communication with the C++ interface objects](#)
2. [Communication with the real-time algorithm](#)

12.8.1 Communication with the C++ interface objects

12.8.1.1 Direct host communication

Most communication between the AAX host and the plug-in is accomplished via the [AAX_IController](#) interface. This interface contains methods for such things as:

- Retrieving environment information such as the current [sample rate](#)
- Getting and setting Effect parameters such as the Effect's [algorithmic delay](#)
- Accessing host-managed information such as [Plug-in meters](#) and MIDI
- Accessing other host-managed communications protocols like [Data packets](#) and MIDI

In addition, the GUI uses a separate interface for managing view and event details with the host. This interface, [AAX_IViewContainer](#), includes methods for:

- Retrieving information like the raw view and the currently held modifier keys
- Requesting changes to view parameters (e.g. size)
- Passing GUI events on to the host.
 - This is an important function because the host may require its own specific behavior for certain events. For example, a command-control-option click in Pro Tools should bring up the parameter's automation menu.

12.8.1.2 Custom data blocks

Often it is necessary to transmit arbitrary blocks of custom plug-in data between different plug-in modules. In AAX, this is accomplished by "pushing" data to and "pulling" it from the plug-in's [data model](#).

The abstract data model interface includes two custom data methods for this:

- [AAX_IEffectParameters::GetCustomData\(\)](#)
- [AAX_IEffectParameters::SetCustomData\(\)](#)

It is the data model's job to act as a go-between when custom data must be transmitted between a plug-in's other modules.

For example, a plug-in may wish to send analysis data from its [direct data module](#) to its [GUI](#). In this situation, the Direct Data object would call [SetCustomData\(\)](#) to update the data model whenever new data was available, while the GUI would "pull" the most up-to-date data via [GetCustomData\(\)](#) whenever an update was required.

Note that the default implementations of these methods are empty and thus all implementation details, including thread safety guards, are left to the plug-in.

12.8.1.3 Notifications

The [data model](#) and [GUI](#) interfaces include [notification hook](#) methods. These methods used for [host-to-Effect notifications](#) by default, but may also be called with custom notification IDs in order to create custom notifications within a plug-in.

12.8.1.4 Direct pointer sharing

If co-location is guaranteed, plug-in modules may directly share data pointers. For example, a non-real-time plug-in's [Host Processor](#) object may share a `this` pointer with its [data model](#) object.

To guarantee co-location between modules that could normally be placed into different memory spaces by the host, use "constraint" properties:

- [AAX_eProperty_Constraint_Location](#)
- [AAX_eProperty_Constraint_Topology](#)

To help avoid forwards-compatibility issues with future devices that support AAX, these constraints should be set whenever a plug-in requires co-location of its components. Note, however, that using a design that relies on co-location will prevent the plug-in from running in distributed environments and should therefore be avoided when possible.

12.8.2 Communication with the real-time algorithm

An AAX plug-in's algorithm is essentially a stateless callback and, therefore, all of its state data must at some level be managed by the host. This model is fundamentally different from the other plug-in modules, which are each objects with their own memory and state.

Most algorithmic data management is performed via the algorithm's context structure. More information about memory management in AAX real-time algorithms can be found [here](#).

12.8.2.1 Data packets

The most common form of communication with a plug-in's real-time algorithm callback is the transmission of read-only data from the data model to the context structure.

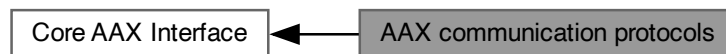
AAX includes a dedicated API for this task that provides buffered, optimized delivery of read-only data packets to the algorithm. For more information, see [Communicating with the algorithm](#).

12.8.2.2 Host-managed context fields

Algorithms can also send data to the host and receive environment information through dedicated context fields. For example, the host can provide access to DMA facilities through an object accessed via a DMA field, and a plug-in can report meter values to the host via a dedicated meter field. For more information, see [Communicating with the algorithm](#).

12.8.2.3 Direct data transfers

When other modules in the plug-in must interact directly with the algorithm's state information this is accomplished via the [Direct Data](#) interface. This interface provides an idle-time context in which the plug-in may read from or write to the algorithm's private data memory. These transfers are unbuffered and therefore the plug-in must handle any appropriate thread-safety considerations. Collaboration diagram for AAX communication protocols:



12.9 AAX Format Specification

Additional requirements for AAX plug-ins.

This document describes aspects of the AAX plug-in format specification that are beyond the scope of the [common interface classes and callbacks](#) that the plug-in must implement.

12.9.1 .aaxplugin Directory Structure

AAX uses a bundle packaging format. On macOS, AAX plug-ins are built as standard OS bundles, while on Windows they are simple directories. All AAX plug-in bindles must use the .aaxplugin extension and the following directory structure:

- /Contents
 - /Resources
 - * *This directory contains all of the additional resource files that will be needed by the plug-in at run time such as DSP algorithm DLLs, XML page tables, and image files for the plug-in's GUI*
 - /MacOS
 - * *Contains the plug-in's macOS binary (Mach-O)**
 - /Win32
 - * *Contains the plug-in's Windows x86 binary**
 - /x64
 - * *Contains the plug-in's Windows x64 binary**
 - /Factory Presets (optional)
 - * *This directory includes built-in plug-in presets. For more information, see [Presets and settings management](#) in the [Pro Tools Guide](#) documentation*
 - PkgInfo (macOS only)
 - * *This file must include the concatenation of the plug-in's CFBundlePackageType (TDMw or BNDL) and CFBundleSignature (PTul)*

- Info.plist (*macOS only*)
 - * *The plug-in's property list*
- desktop.ini (Windows only)
 - *The .aaxplugin directory's view resource file, used to set its custom icon in Windows Explorer*
- PlugIn.ico (Windows only)
 - *Custom plug-in icon file*

*See the following compatibility notes

Host Compatibility Notes

- The plug-in's binary filename must be the same as the outer .aaxplugin bundle name

Host Compatibility Notes

- On Windows, the plug-in binary (DLL) must use the ".aaxplugin" suffix; i.e. the DLL must use exactly the same name as the outer .aaxplugin folder. On macOS, the plug-in binary does not require a specific suffix.

Host Compatibility Notes

- On Windows, the plug-in's binary filename (and therefore also the outer .aaxplugin file name) must not contain any spaces. There is a bug in AAE that will prevent binaries with spaces from being loaded properly. This is logged as PTSW-189928.

Note

This directory structure is also used for plug-in installer directories in the VENUE plug-in installer system. See [VENUE Plug-in installer specification](#) for more information.

12.9.2 Required Symbols

The following symbols are required in any AAX plug-in and must not be stripped from the binary:

- _ACFRegisterPlugin
- _ACFRegisterComponent
- _ACFGetClassFactory
- _ACFCanUnloadNow
- _ACFStartup
- _ACFShutdown
- _ACFGetSDKVersion *

Host Compatibility Notes * _ACFGetSDKVersion is required for 64-bit AAX plug-ins only

Collaboration diagram for AAX Format Specification:



12.10 Additional AAX features

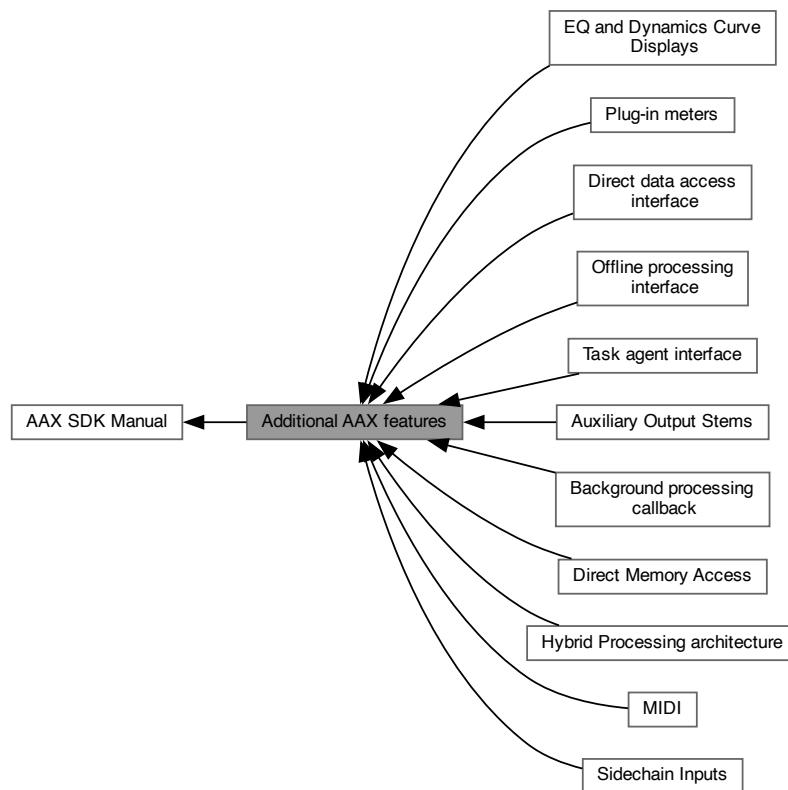
12.10.1

How to use additional features and functionality supported by AAX.

Documents

- [Direct data access interface](#)
A host interface providing direct access to a plug-in's algorithm memory.
- [Offline processing interface](#)
Advanced offline processing features.
- [Hybrid Processing architecture](#)
An architecture combining low-latency and high-latency audio processing.
- [MIDI](#)
How to route and process MIDI in AAX plug-ins.
- [Plug-in meters](#)
How to manage metering data for AAX plug-ins.
- [Sidechain Inputs](#)
Routing custom audio streams to a plug-in.
- [Auxiliary Output Stems](#)
Routing custom audio streams from a plug-in.
- [Direct Memory Access](#)
DMA support for AAX DSP plug-ins, with emulation for AAX Native.
- [Background processing callback](#)
Background processing support for AAX DSP and Native plug-in algorithms.
- [EQ and Dynamics Curve Displays](#)
Displaying EQ and Dynamics curves in Pro Tools, control surfaces, and other auxiliary graphical interfaces.
- [Task agent interface](#)
A mechanism for hosts to request that plug-ins perform tasks.

Collaboration diagram for Additional AAX features:



12.11 Direct data access interface

12.11.1

A host interface providing direct access to a plug-in's algorithm memory.

This interface represents an optional component that you can add to your plug-in in order to support extended features of the AAX SDK.

Some plug-ins require the host to retrieve non-meter data from the decoupled algorithm module to display on a GUI or perform additional computation. For example, the result of computing the audio spectrum or pitch data in the algorithm can be delivered to the host to display on-screen. This is the purpose of the [AAX_IEffectDirectData](#) interface.

The [Direct Data interface](#) provides facilities for directly accessing a plug-in's algorithm memory. This interface may be used to transfer private data from the algorithm to other plug-in components, such as the [GUI](#). It may also be used as an alternative to [PostPacket\(\)](#) to perform direct writes to the algorithm's private data memory.

To set up Direct Data, the module must be registered with the host in the plug-in's Description callback like other process pointers. To add this interface to your plug-in at describe time, call [AAX_IEffectDescriptor::AddProcPtr\(\)](#) using the [kAAX_ProcPtrID_Create_EffectDirectData](#) selector.

The DirectData module works for all plug-in types, including AAX Native, AAX DSP, and AAX AudioSuite.

12.11.2 Convenience class

[AAX_CEffectDirectData](#), the concrete implementation of [AAX_IEffectDirectData](#), consists of a [TimerWakeup_PrivateDataAccess\(\)](#) function that you subclass in order to access an algorithm's private state data. This timer wakes up at a periodic interval. In this function you can read the algorithm's private data port to pull the state of an algorithm. Note that the wakeup period is variable depending on the plug-in's buffer size and running context (real time processing, AudioSuite, offline bounce, etc.) Care must be taken to ensure that any data retrieved from the algorithm is either buffered to handle the thread callback periods for the various running contexts or that the plug-in does not depend on the Direct Data timer catching every state update.

[AAX_CEffectDirectData](#) also includes convenience accessors to the Controller and Data Model in order to help facilitate common access scenarios. Using these, you can do any computation necessary to handle the incoming algorithm state data and send results on to the Data Model and/or the GUI interface.

12.11.3 Private data access interface

The Direct Data API provides a [TimerWakeup](#) callback with access to [AAX_IPrivateDataAccess](#). This reference is only valid within the context of the wakeup callback and cannot be stored to provide private data access in other contexts.

The Private Data Access interface can be used to directly read from and write to an algorithm's private data. These operations are not synchronized with the algorithm's processing callback, which may asynchronously pre-empt the read or write operations. Plug-ins that use this interface should buffer all access to their private data to ensure data integrity.

12.11.4 Communicating with other modules

The Direct Data API does not include any facilities for inter-module communication. In order to transfer data between a plug-in's [AAX_IEffectDirectData](#) object and its other objects, dedicated custom data methods in those objects' interfaces should be used. For example, to communicate with the plug-in's data model, use [AAX_IEffectParameters::GetCustomData\(\)](#) and [AAX_IEffectParameters::SetCustomData\(\)](#)

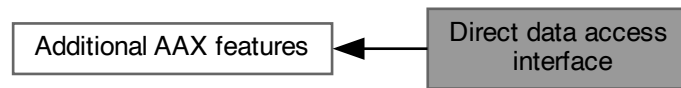
See also

[Hybrid Processing architecture](#) for another approach to transferring large amounts of (audio) data between the algorithm callback and the plug-in's data model.

Classes

- class [AAX_CEffectDirectData](#)
Default implementation of the [AAX_IEffectDirectData](#) interface.
- class [AAX_IACFEffectDirectData](#)
Optional interface for direct access to a plug-in's alg memory.
- class [AAX_IACFPrivateDataAccess](#)
Interface for the AAX host's data access functionality.
- class [AAX_IEffectDirectData](#)
The interface for a AAX Plug-in's direct data interface.
- class [AAX_IPrivateDataAccess](#)
Interface to data access provided by host to plug-in.

Collaboration diagram for Direct data access interface:



12.12 Offline processing interface

12.12.1

Advanced offline processing features.

This interface represents an optional component that you can add to your plug-in in order to support extended features of the AAX SDK.

The HostProcessor interface provides offline plug-ins with useful offline processing features such as random-access facilities and a non-processing analysis callback. For documentation, see the following classes:

- [Host processor module](#)
- [Host processor delegate](#)

To add this interface to your plug-in at describe time, register a [ProcPtr](#) using the [kAAX_ProcPtrID_Create_HostProcessor](#) selector.

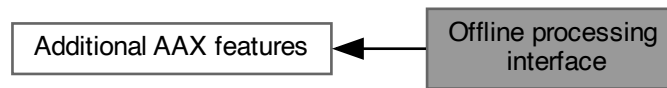
Note

If your plug-in does not require the specific offline processing features provided by this interface then it should not register a host processor. Instead, register an offline version of the plug-in's real-time algorithm using the [AAX_eProperty_PluginID_AudioSuite](#) property.

Classes

- class [AAX_CHostProcessor](#)
Concrete implementation of the [AAX_IHostProcessor](#) interface for non-real-time processing.
- class [AAX_IACFHostProcessor](#)
Versioned interface for an AAX host processing component.
- class [AAX_IHostProcessor](#)
Base class for the host processor interface.
- class [AAX_IHostProcessorDelegate](#)
Versioned interface for host methods specific to offline processing.
- class [AAX_VHostProcessorDelegate](#)
Version-managed concrete [Host Processor delegate](#) class.

Collaboration diagram for Offline processing interface:



12.13 Hybrid Processing architecture

12.13.1

An architecture combining low-latency and high-latency audio processing.

12.13.2 Overview of Hybrid

Hybrid processing is an optional feature that allows a single plug-in to simultaneously render data on the host's low- and high-latency signal networks. In many large plug-ins this can be very useful. For example, consider a reverb algorithm with both early reflection and tail processing. With AAX Hybrid, this plug-in can process the early reflections at low latency while allowing the tail algorithm to be handled at higher latency (and thus higher efficiency.) Other kinds of algorithms that could benefit from Hybrid processing are noise reductions, analyzers, multi-effect suites, and instruments.

Because the low-latency AAX signal network may be run on DSP hardware, AAX DSP plug-ins that incorporate Hybrid processing can split audio processing between the DSP and the host. This provides the benefits of low latency, highly deterministic DSP-based processing while also allowing the plug-in to leverage the high-latency power of the Intel core where appropriate.

AAX Hybrid is an internal feature and is not exposed to users, except in terms of better plug-in performance and more efficient DSP usage.

Note

AAX Hybrid may be protected by one or more U.S. and non-U.S. patents. Details are available at www.avid.com/patents.

12.13.3 Implementing Hybrid processing

For an example of Hybrid processing, see the [DemoDelay_Hybrid](#) example plug-in

To register for Hybrid processing, a plug-in should add values for [AAX_eProperty_HybridInputStemFormat](#) and [AAX_eProperty_HybridOutputStemFormat](#) to the associated ProcessProc property map. Once these values have been registered, both the ProcessProc callback and the Hybrid render function in the plug-in's data model will be invoked during processing.

Hybrid processing context information is provided via a dedicated [Hybrid processing context structure](#). It is not possible to register additional fields on this context. However, unlike a normal algorithm `ProcessProc`, the Hybrid render method is implemented directly within the plug-in's effect parameters object and has direct access to the data model memory. This is possible since the render method will always run on the host, and makes it easier to implement algorithms that require access to the data model, e.g. for direct access to impulse responses, etc.

The AAX host provides dedicated audio buffers in both the `ProcessProc` context and the Hybrid processing context. These buffers can be used to pass audio data between the low-latency `ProcessProc` and the Hybrid render contexts.

- The plug-in may pass output from the low-latency `ProcessProc` to the Hybrid render method using additional audio buffers that are added at the end of the `ProcessProc` context's normal output buffer array. The `ProcessProc` may perform any pre-processing that is desired before passing audio to the Hybrid render context via these buffers. The [AAX_eProperty_HybridOutputStemFormat](#) property defines how many buffers will be sent from the `ProcessProc` to the Hybrid render method.
- Similarly, the plug-in may pass samples from the Hybrid processing callback to the low-latency `ProcessProc` using additional audio buffers that are added at the end of the `ProcessProc` context's normal input buffer array. The [AAX_eProperty_HybridInputStemFormat](#) property defines how many buffers will be sent from the Hybrid render method to the `ProcessProc`.

Samples which are sent from the `ProcessProc` to the Hybrid processing callback and back to the `ProcessProc` are delayed by a fixed amount relative to the normal input samples that are processed directly by the `ProcessProc` to its output buffers. The number of samples of delay that are added in this round-trip is available to the plug-in via [AAX_IController::GetHybridSignalLatency\(\)](#).

12.13.4 Additional information

12.13.4.1 Parameter update timing

Because updates are not passed to the [Hybrid processing context](#) using the normal AAX port infrastructure, any parameter updates from automation will be reflected in this context a little bit ahead of time (~21 ms at 44.1 kHz.) See the [Parameter update timing](#) page for a discussion of parameter timing accuracy and some suggestions of how you can maintain accurate parameter update timing.

12.13.4.2 Host support and alternatives

Not all [AAX](#) hosts support [AAX](#) Hybrid processing. See the [Host Support](#) page for additional information.

See also

[Direct data access interface](#) for another approach for transferring non-audio data between the algorithm callback and the plug-in's data model.

Classes

- struct [AAX_SHybridRenderInfo](#)
Hybrid render processing context.

Hybrid audio methods

- virtual [AAX_Result AAX_IACFEffParameters_V2::RenderAudio_Hybrid](#) ([AAX_SHybridRenderInfo](#) *ioRenderInfo)=0
Hybrid audio render function.

MIDI methods

Methods to access the plug-in's host-managed MIDI information.

- virtual [AAX_Result AAX_IController::GetHybridSignalLatency](#) (int32_t *outSamples) const =0
CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.

12.13.5 Function Documentation

12.13.5.1 RenderAudio_Hybrid()

```
virtual AAX\_Result AAX_IACFEffParameters_V2::RenderAudio_Hybrid (
    AAX\_SHybridRenderInfo * ioRenderInfo ) [pure virtual]
```

Hybrid audio render function.

This method is called from the host to render audio for the hybrid piece of the algorithm.

Note

To use this method plug-in should register some hybrid inputs and outputs in "Describe"

Implemented in [AAX_CEffectParameters](#).

12.13.5.2 GetHybridSignalLatency()

```
virtual AAX\_Result AAX_IController::GetHybridSignalLatency (
    int32_t * outSamples ) const [pure virtual]
```

CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.

This method provides the number of samples that the AAX host expects the plug-in to delay a signal. The host will use this value when accounting for latency across the system.

Note

This value will generally scale up with sample rate, although it's not a simple multiple due to some fixed overhead. This value will be fixed for any given sample rate regardless of other buffer size settings in the host app.

Parameters

out	<i>outSamples</i>	The number of samples of hybrid signal delay
-----	-------------------	--

Implemented in [AAX_VController](#).

Collaboration diagram for Hybrid Processing architecture:



12.14 MIDI

How to route and process MIDI in AAX plug-ins.

12.14.1 Midi Overview

DirectMidi is Avid's protocol for communication of MIDI and other timing-critical plug-in information. It is a cross-platform solution to tightly integrate the host application, audio engine, and plug-ins.

12.14.2 MIDI node types

There are four kinds of nodes an AAX plug-in can create. See [AAX_eMIDINodeType](#) for additional details about these node types:

- [AAX_eMIDINodeType_LocalInput](#)
- [AAX_eMIDINodeType_LocalOutput](#)
- [AAX_eMIDINodeType_Global](#)
- [AAX_eMIDINodeType_Transport](#)

12.14.3 Adding MIDI functionality to a plug-in

Plug-in may access MIDI data in its algorithm or data model. If plug-in needs MIDI in both places or just in the algorithm, it should add a MIDI node to the algorithm context, i.e. call `AAX_IComponentDescriptor::AddMIDINode()` with the appropriate node type.

```
//=====
// Algorithm context definitions
//=====

// Context structure
struct SMY_AlgorithmContext
{
    [...]
    AAX_IMIDINode * mMIDIInNodeP;           // Local input MIDI node pointer
    AAX_IMIDINode * mMIDIOutNodeP;          // Local output MIDI node pointer
    AAX_IMIDINode * mMIDITransportNodeP;    // Transport node
    [...]
};

enum EDemoMIDI_AlgorithmPortID
{
    [...]
    //
    // Add the MIDI node as a physical address within the context field
    ,eAlgorithmPortID_MIDIInNodeA           = AAX_FIELD_INDEX (SDemoMIDI_AlgorithmContext, mMIDIInNodeP)
    ,eAlgorithmPortID_MIDIOutNodeA          = AAX_FIELD_INDEX (SDemoMIDI_AlgorithmContext, mMIDIOutNodeP)
    ,eAlgorithmPortID_MIDITransportNodeA    = AAX_FIELD_INDEX (SDemoMIDI_AlgorithmContext, mMIDITransportNodeP)
    [...]
};

// *****
// ROUTINE: DescribeAlgorithmComponent
// Algorithm component description
// *****
static void DescribeAlgorithmComponent( AAX_IComponentDescriptor * outDesc )
{
    AAX_Result err;

    [...]
    // Register MIDI nodes
    err = outDesc->AddMIDINode(eAlgorithmPortID_MIDIInNodeA, AAX_eMIDINodeType_LocalInput, "DemoMIDI", 0xffff);
    AAX_ASSERT (err == 0);
    err = outDesc->AddMIDINode(eAlgorithmPortID_MIDIOutNodeA, AAX_eMIDINodeType_LocalOutput, "DemoMIDIOut",
        0xffff); AAX_ASSERT (err == 0);
    err = outDesc->AddMIDINode(eAlgorithmPortID_MIDITransportNodeA, AAX_eMIDINodeType_Transport, "DemoMIDITransprt",
        0xffff); AAX_ASSERT (err == 0);
    [...]
}
```

If MIDI data is needed in the plug-in's data model only, plug-in should describe MIDI node with `AAX_IEffectDescriptor::AddControlMIDINode()`

```
// *****
// ROUTINE: GetPlugInDescription
// *****
static AAX_Result GetPlugInDescription( AAX_IEffectDescriptor * outDescriptor )
{
    AAX_Result err;

    [...]
    // Register MIDI nodes
    err = outDescriptor->AddControlMIDINode('linp', AAX_eMIDINodeType_LocalInput, "DemoMIDI", 0xffff);
    AAX_ASSERT (err == 0);
    err = outDescriptor->AddControlMIDINode('lout', AAX_eMIDINodeType_LocalOutput, "DemoMIDIOut", 0xffff);
    AAX_ASSERT (err == 0);
    err = outDescriptor->AddControlMIDINode('tran', AAX_eMIDINodeType_Transport, "DemoMIDITransprt", 0xffff);
    AAX_ASSERT (err == 0);
    [...]

    return err;
}
```

Note

These two types of MIDI nodes can't be used together in the same plug-in's effect.

12.14.4 Using MIDI in a plug-in algorithm

Like with other algorithm context ports, data in MIDI nodes is directly available in the plug-in's algorithm process function. Here is an example from the DemoMIDI_NoteOn sample plug-in:

```

template<int kNumChannelsIn, int kNumChannelsOut>
void
AAX_CALLBACK
DemoMIDI_AlgorithmProcessFunction (
    SDemoMIDI_Alg_Context * const    inInstancesBegin [],
    const void *                inInstancesEnd)
{
    [...]
    // Setup MIDI In node pointers
    AAX_IMIDINode* midiNodeIn = instance->mMIDINodeP;
    AAX_CMidiStream* midiBufferIn = midiNodeIn->GetNodeBuffer();
    AAX_CMidiPacket* midiBufferInPtr = midiBufferIn->mBuffer;
    uint32_t packets_count_in = midiBufferIn->mBufferSize;

    // Setup MIDI Out node pointers
    AAX_IMIDINode* midiNodeOut = instance->mMIDINodeOutP;
    AAX_CMidiStream* midiBufferOut = midiNodeOut->GetNodeBuffer();
    AAX_CMidiPacket* midiBufferOutPtr = midiBufferOut->mBuffer;
    uint32_t packets_count_out = midiBufferOut->mBufferSize;

    // Setup MIDI Transport node pointers
    // NOTE: See warning at AAX_IMIDINode::GetTransport() regarding use of this interface
    AAX_IMIDINode* midiTransport = instance->mMIDINodeTransportP;
    AAX_ITransport * transport = midiTransport->GetTransport();
    bool transport_is_playing = false;
    if (transport) {
        transport->IsTransportPlaying(&transport_is_playing);
    }

    if(transport_is_playing) {
        //
        // While there are packets in the node
        while (packets_count_in > 0) {
            midiBufferOutPtr = midiBufferInPtr;           // Copy the packet from the input MIDI node
                                                         // to the output MIDI node
            midiBufferOutPtr->mTimestamp = timeStamp;      // Set the MIDI time stamp
            midiNodeOut->PostMIDIPacket(midiBufferOutPtr); // Post the MIDI packet
            midiBufferOut->mBufferSize = packets_count_in;

            midiBufferInPtr++;
            packets_count_in--;
        }
    }
    [...]
}

```

Also data from the MIDI nodes that were described with [AAX_IComponentDescriptor::AddMIDINode\(\)](#) can be accessed via [AAX_CEffectParameters::UpdateMIDINodes\(\)](#) method. This method provides an [AAX_CMidiPacket](#). Because the MIDI packet structure does not identify the associated MIDI stream's type (input, output, global, or transport) this method also provides an index into the plug-in's algorithm context structure which can be used to identify the semantics of the MIDI packet.

12.14.5 Accessing MIDI in the plug-in data model

A plug-in may access MIDI data in its data model via the [AAX_CEffectParameters::UpdateMIDINodes\(\)](#) or [AAX_CEffectParameters::UpdateControlMIDINodes\(\)](#) methods. Both of these methods provide an [AAX_CMidiPacket](#). Because the MIDI packet structure does not identify the associated MIDI stream's type (input, output, global, or transport) UpdateMIDINodes method also provides an index into the plug-in's algorithm context structure which can be used to identify the semantics of the MIDI packet, while UpdateControlMIDINodes provides MIDI node ID for the same reason.

```

AAX_Result DemoMIDI_Parameters::UpdateMIDINodes ( AAX_CFieldIndex inFieldIndex,    AAX_CMidiPacket& inPacket
)
{
    if (eAlgPortID_MIDINodeIn == inFieldIndex)
    {
        if ( (inPacket.mData[0] & 0xF0) == 0x90 )
        {
            if ( inPacket.mData[2] == 0x00 )
            {
                // Note Off
            }
            else
            {
                // Note On
            }
        }
    }
}

```

```

    }

    return AAX_SUCCESS;
}

```

Note

Only one of the `UpdateMIDINodes` and `UpdateControlMIDINodes` can be used in the single plug-in's effect at a time. If plug-in uses MIDI nodes described with `AddMIDINode` function, then only `UpdateMIDINodes` method can be used to receive MIDI messages. Otherwise `UpdateControlMIDINodes` should be used.

Collaboration diagram for MIDI:



12.15 Plug-in meters

How to manage metering data for AAX plug-ins.

12.15.1 Overview of metering in AAX

AAX provides a host-managed metering system for plug-ins. The host buffers, thins, and applies ballistics to each of the plug-in's meters. When the plug-in GUI retrieves this processed data, it receives the exact same information that is displayed on control surfaces and other metering devices.

12.15.2 Adding meters to an Effect

Meters are added to an algorithm Component in [Describe](#) using `AAX_IComponentDescriptor::AddMeters()`. The resulting meter context field will be populated with an array of meter "tap" values, one for each of the Component's meters.

12.15.2.1 Customizing meter behavior

Using the [Effect Descriptor](#), each meter in the Effect may optionally be associated with a [property map](#) that applies a particular set of display properties to the meter. These are the properties that may be set on a meter:

- [AAX_EMeterOrientation](#)
- [AAX_EMeterBallisticType](#)
- [AAX_EMeterType](#)

Note that, because meter properties are added at the Effect level, it is not possible to describe different meter property configurations for different algorithms in the same Effect.

12.15.3 Reporting meter values

Meter values are reported by the algorithm using one "tap" per channel per buffer. For each tap, the algorithm must report the maximum metered sample value for each processing buffer.

Meter tap values can be interpreted as the maximum value of the meter per buffer, on a scale of [0.0 1.0]. In all cases the plug-in's meter position should be normalized between 0 and 1, where 0 is no gain reduction. For example:

- An input meter should report the maximum absolute sample value that is present in the input audio buffer for the appropriate channel
- An output meter should report the maximum absolute sample value that is present in the output audio buffer for the appropriate channel
- A gain-reduction meter (CL or EG types) should report the largest amount of gain reduction in the current buffer for the appropriate channel. If no gain reduction occurred for a buffer then a value of 0.0 should be reported. If a full-scale signal was reduced to silence then a value of 1.0 should be reported.

Gain-reduction meter values should report peak gain reduction, not RMS or other algorithms, and may use any normalization mapping (e.g. linear, exponential) which is desired. Ideally the gain-reduction metering UI in the host and on attached control surfaces will match the Peak gain reduction metering in the plug-in's GUI.

Legacy Porting Notes The gain-reduction meter handling for AAX plug-ins is different from that for RTAS/TDM plug-ins. AAX plug-ins must invert their gain-reduction meter values manually before reporting these values from the audio processing callback. The AAX host will always thin reported meter data using a "max" operation, and will later invert gain-reduction meter values before they are available to the plug-in GUI or to control surfaces.

12.15.4 Displaying meter values

The meter values that are reported to the system from the algorithm are available, in buffered and (optionally) ballistics-smoothed form, from [AAX_IController](#). The meter values returned from methods such as [GetCurrentMeterValue\(\)](#) and [GetMeterPeakValue\(\)](#) are the same values used by the system when displaying plug-in meters on control surfaces, and when a plug-in clears the peak value using [ClearMeterPeakValue\(\)](#) this change will likewise be reflected throughout the system.

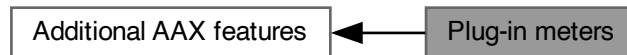
The literal values provided by these methods can be interpreted as the distance from "rest" that the meter must travel to represent the current value, again on a scale of [0.0 1.0]. Note that this is not necessarily equivalent to the semantics of the meter's reported values in the algorithm:

- For "standard" meters such as input meters, this corresponds to the value provided by the algorithm, since a maximum metered sample value (1.0) corresponds to a meter that should be drawn "furthest from rest" (1.0), i.e. at the top of a standard bottom-to-top meter graphic, or at the far right of a standard left-to-right graphic.
- For "inverted" meters, such as gain-reduction meters, these semantics are reversed: a maximum metered sample value (1.0) corresponds to a meter drawn "at rest" (0.0), i.e. at the bottom of a bottom-to-top meter graphic or at the far left of a left-to-right graphic.

These values are independent of [meter orientation](#): an input or output meter that is oriented with [AAX_eMeterOrientation_TopRight](#) will still use 0.0 as its "at rest" position, and likewise a gain-reduction meter that is oriented with [AAX_eMeterOrientation_BottomLeft](#) will still use 1.0.

12.15.5 Alternatives

For advanced metering applications a single tap value may not be sufficient. To transmit more detailed information from the algorithm to its other components, a plug-in must use the [Direct Data](#) interface. Collaboration diagram for Plug-in meters:



12.16 Sidechain Inputs

Routing custom audio streams to a plug-in.

12.16.1 Overview of Sidechain Inputs

If applicable, plug-ins may choose to enable sidechain inputs. If a sidechain is enabled, a menu is added to the plug-in's header that allows the user to choose an interface or bus as the sidechain, or "key input". Once enabled, the plug-in will be able to access sidechain input just like any other input signal. Currently, DAE is limited to mono sidechain inputs.

12.16.2 Adding a Sidechain Input to an Effect

Setting up a sidechain input is fairly straight forward. You will want to add a physical address within your context structure, and then "describe" the sidechain in Describe.

Context Structure:

```

//=====
// Component context definitions
//=====

// Context structure
struct SMyPlugIn_Alg_Context
{
    [...]
    int32_t * mSideChainP;
    [...]
};

// Physical addresses within the context
enum EDemoDist_Alg_PortID
{
    [...]
    ,MyPlugIn_AlgFieldID_SideChain = AAX_FIELD_INDEX (SDemoDist_Alg_Context, mSideChainP)
    [...]
};
  
```

Describe:

```

// *****
// ROUTINE: DescribeAlgorithmComponent
// Algorithm component description
// *****
static void DescribeAlgorithmComponent( AAX_IComponentDescriptor * outDesc )
  
```

```

{
    AAX_Result          err = AAX_SUCCESS;

    [...]
    err = outDesc.AddSideChainIn(eDemoDist_AlgFieldID_SideChain);
    [...]
    properties->AddProperty ( AAX_eProperty_SupportsSideChainInput, true );
    [...]
}

```

Todo Is `properties->AddProperty (AAX_eProperty_SupportsSideChainInput, true)` even necessary?!?! I believe I saw a p.i. that does not declare this...

In order to tell whether there is sidechain information available to your plug-in, check for a null pointer within your algorithm's process function. The sidechain channel will show up as an additional stem from the original stem format you declare. That is to say, for a stereo plug-in, the sidechain channel will be the third channel passed in.

```

//=====
// Processing function definition
//=====

void
AAX_CALLBACK
MyPlugIn_AlgorithmProcessFunction (
    SMyPlugIn_Alg_Context * const    inInstancesBegin [],
    const void *                  inInstancesEnd)
{
    [...]
    int32_t sideChainChannel = *instance->mSideChainP;
    float * AAX_RESTRICT sideChainInput = 0;
    if ( sideChainChannel )
        sideChainInput = instance->mInputPP [sideChain]Channel;
    [...]
}

```

Collaboration diagram for Sidechain Inputs:



12.17 Auxiliary Output Stems

Routing custom audio streams from a plug-in.

12.17.1 Overview of Auxiliary Output Stems in AAX

Pro Tools has the capability to show and route multiple "auxiliary" outputs from a plug-in to other tracks. These are known as Auxiliary Output Stems (AOS), a stem referring to one set of outputs. A stereo stem contains two outputs, left and right, and a mono stem contains one output. The outputs will appear in the input assignment pop-up menu of each track under the category "plug-in".

Your plug-in is responsible for the definition of valid aux output paths. This definition includes the total number of outputs and the desired order of stereo and mono paths. Pro Tools will query each plug-in for available valid paths and populate its track input selector popup menus accordingly.

Plug-ins must define the lowest available aux output number. In other words, the port number of an aux output needs to be the lowest available port number after the main outputs of the track the plug-in is instantiated on. For example, the first available aux output for the plug-in residing on a 5.1 surround track would have a port number of 7, since there are 6 main outputs for the track.

Additionally, port numbers must be declared sequentially and in the order aux output stems are added. For example, a stem cannot be added with the port number 10 if it precedes a stem with the port number 4.

12.17.2 Implementing Auxiliary Output Stems

The Auxiliary Output Stems API has a specific descriptor associated with it that needs to be added in `DescribeAlgorithmComponent::AddAuxOutputStem()`. Make sure this method is called for each component that supports a different stem format. For example, a mono aux output would be defined as follows:

```
// *****
// ROUTINE: DescribeAlgorithmComponent
// Algorithm component description
// *****
static void DescribeAlgorithmComponent( AAX_IComponentDescriptor * outDesc )
{
    AAX_Result err = AAX_SUCCESS;

    [...]
    err = outDesc->AddAuxOutputStem(0 /* first parameter is not used */,
                                     AAX_eStemFormat_Mono,
                                     "My Auxiliary Output Channel");
    AAX_ASSERT (err == AAX_SUCCESS);
    [...]
}
```

The auxiliary output buffers for the plug-in will be appended to the normal output buffer array in the plug-in algorithm.

Warning

Some hosts, such as Media Composer, do not support Auxiliary Output Stems. You must clearly document that your plug-ins are not supported on these hosts; attempts by the plug-in to write data beyond the end of the audio output buffer may cause crashes and other bugs in these hosts. See [Host Support](#) for more information.

In your plug-in's algorithm, you will simply need to account for the extra outputs when it processes the audio. Pro Tools will not automatically route your processed audio to all the extra outputs. As with main outputs, make sure the processed audio samples are placed in the auxiliary outputs' buffers as well. Collaboration diagram for Auxiliary Output Stems:



12.18 Direct Memory Access

DMA support for AAX DSP plug-ins, with emulation for AAX Native.

12.18.1 On this page

- [DMA facility overview](#)
- [DMA transfer modes](#)
- [Registering for DMA transfers](#)
- [DMA restrictions](#)
- [Additional information](#)

12.18.2 DMA facility overview

AAX provides an [abstract interface](#) for accessing the host environment's DMA or other memory-transfer facilities. All platform-specific details are handled by the AAX host environment, allowing plug-ins that use this interface to be re-targeted to Native or DSP environments without changing their memory transfer implementation.

12.18.3 DMA transfer modes

AAX hosts may support the following DMA modes, as listed in [AAX_IDma::EMode](#) :

- In [Scatter](#) mode, data is transferred from a linear buffer to a series of offset segments in a circular buffer. This mode is most often used to transfer data from linear internal memory to a large external memory buffer.
- In [Gather](#) mode, data is collected from a series of offset segments in a circular buffer and concatenated in a linear buffer. This mode is most often used to transfer data from an external memory buffer to an internal memory buffer.
- In [Burst](#) mode, data is written linearly from one location to another. Burst mode transfers may be used for linear transfers of data to or from external memory. During the transfer, the source data is broken into a series of individual bursts. This mode is included for completeness, though the Scatter/Gather modes are expected to be more appropriate for the vast majority of real-world DMA use cases.

12.18.4 Registering for DMA transfers

Algorithm Components register for DMA transfers by adding one or more DMA fields to their context via [AAX_IComponentDescriptor::AddDmaInstance\(\)](#). At runtime, each field will be populated with a valid [DMA interface](#) for the specified DMA mode.

12.18.5 DMA restrictions

The following restrictions apply to DMA transfers on all AAX platforms:

- The maximum burst size for any DMA transfer is 64B. The minimum burst size is 1B.
- Only one DMA transfer request may be posted per [AAX_IDma](#) object per processing callback.
- Scatter and Gather requests each require that the circular memory buffer be padded by at least the size of one burst

12.18.6 Additional information

HDX DSP Guide

- [DMA support](#)
- [DMA and background thread performance reporting](#)

Collaboration diagram for Direct Memory Access:



12.19 Background processing callback

Background processing support for AAX DSP and Native plug-in algorithms.

12.19.1 On this page

- [Background thread description](#)
- [Restrictions and limitations of background threads](#)
- [Background thread performance characteristics on DSP systems](#)
- [Background thread memory management](#)
- [Additional information](#)

12.19.2 Background thread description

Each algorithm render callback may optionally be associated with a background processing callback. This background callback will be triggered regularly in an idle context on a separate thread, and can be used to perform any background task required by the algorithm.

Background thread processing is supported for both AAX DSP and AAX Native plug-ins.

12.19.3 Restrictions and limitations of background threads

- An AAX DSP Effect that registers for background processing will not share a DSP with any other Effect type. It may share a DSP with multiple instances of its own type, but only if its resource requirements allow for this.
- The frequency of background thread executions relative to render thread executions will vary depending on the processing situation. For example, a host may pre-process a series of audio buffers as quickly as possible during an offline render. In this case there would be many executions of the render thread callback for each execution of the background thread callback. Be sure to consider this when using the background thread feature in plug-ins that support an AudioSuite processing type.

12.19.4 Background thread performance characteristics on DSP systems

The background processing callback is called from a true idle thread context. On DSP accelerated platforms, this means that the callback will be triggered continuously whenever the chip is not executing an interrupt, i.e. the algorithm render callback. Since the render callback's resource requirements are well-defined (or at least strictly bounded,) the background thread's available cycles are also deterministically bounded.

However, the background thread itself has a lower priority than the DSP shell. While the background callback's execution will not be interrupted by shell operations, it will be blocked in the event of a contention for memory resources with the shell. As a result, the number of memory operations that may be performed in this callback will be less well-defined when the host is consuming memory resources, e.g. when delivering a very large coefficient block to the DSP.

If your TDM plug-in does not perform any resource-intensive memory operations then you can assume a guaranteed performance level for its DSP background thread. Development tools are available that will test a plug-in by refreshing its entire context memory at every interrupt, and the background thread performance characteristics measured by these tools, plus an additional buffer to account for any pathological cases that may be missed by the performance check, should provide a guaranteed performance baseline for the background thread that will be completely safe for any Pro Tools operation scenario.

12.19.5 Background thread memory management

The background processing callback is not provided with any data pointers and does not have access to any facilities for managed communication with the rest of the plug-in. Therefore, the background process must use shared global data structures to interact with the render callback. Your plug-in will need to manually synchronize access to this data.

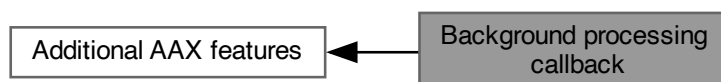
Usually the background callback will want to interact with the render callback via the algorithm's private data blocks. Therefore, private data blocks that are provided to an algorithm's context will not be relocated by the host between calls to the render callback, and background processes can reliably access this data once provided with a pointer. The same is not true for audio buffers, meters, coefficient ports, etc. - this data can all be relocated by the host when the render callback is not executing.

12.19.6 Additional information

HDX DSP Guide

- [Background processing](#)
- [DMA and background thread performance reporting](#)

Collaboration diagram for Background processing callback:



12.20 EQ and Dynamics Curve Displays

12.20.1

Displaying EQ and Dynamics curves in Pro Tools, control surfaces, and other auxiliary graphical interfaces.

About Pro Tools, control surfaces, and other auxiliary displays connected to the AAX host may provide a curve data display to enhance the graphical representation of the plug-in's state.

A "bouncing ball" meter may also be overlaid within the curve data presentation for Dynamics plug-ins.

Pro Tools Mix Window displaying EQ plug-in instances

Pro Tools | S6 MTM display showing a Dynamics plug-in instance with bouncing-ball metering

Requirements

Host Compatibility Notes For S6 control surface displays, see [PT-226228](#) and [PT-226227](#) in the [Known Issues](#) page for more information about the requirements listed in this section.

These are the requirements for supporting the AAX curve data display features:

- To support EQ curve data displays, a plug-in must support [AAX_IEffectParameters::GetCurveData\(\)](#)
- To support Dynamics curve data displays, a plug-in must also support [AAX_IEffectParameters::GetCurveDataDisplayRange\(\)](#) for the Dynamics curve data types.

The AAX host will only query and display curve data for plug-ins of the applicable [Category](#), as specified in the [AAX_ECurveType](#) documentation.

In order to present a bouncing-ball metering display, Dynamics plug-ins must also support [AAX_IEffectParameters::GetCurveDataMetering\(\)](#) in addition to the two other curve data methods. This feature is always optional: a Dynamics plug-in may present a curve only without support for a bouncing-ball meter overlay.

Pro Tools Implementation There are three different kinds of calls that Pro Tools will make when querying EQ plug-ins for curve data:

- An initial query with a small set of points. This query is used only to determine whether the plug-in supports the EQ Curve display feature. The result data is not used. If the plug-in does not support the feature it must return an error value from [GetCurveData\(\)](#)
- A normal full-curve query to get the base curve data across the full display range
- One or more targeted queries around any detected inflection points. These queries are used to increase display resolution for plug-ins with very narrow Q settings.

Pro Tools will call [GetCurveData\(\)](#) from a thread in a low-priority thread pool. Most other Pro Tools operations will not be blocked by the execution of this method, though note that if a control surface is also issuing queries then the method may be called concurrently from multiple host threads.

In Pro Tools, curve updates are triggered by incrementing the plug-in's change counter. This is the counter value returned from the [GetNumberOfChanges\(\)](#) method in the plug-in. This counter is updated automatically when any plug-in parameter changes.

Enumerator

AAX_eCurveType_None	
AAX_eCurveType_EQ	EQ Curve, input values are in Hz, output values are in dB. Host Compatibility Notes Pro Tools requests this curve type for EQ plug-ins only
AAX_eCurveType_Dynamics	Dynamics Curve showing input vs. output, input and output values are in dB. Host Compatibility Notes Pro Tools requests this curve type for Dynamics plug-ins only
AAX_eCurveType_Reduction	Gain-reduction curve showing input vs. gain reduction, input and output values are in dB. Host Compatibility Notes Pro Tools requests this curve type for Dynamics plug-ins only

12.20.3 Function Documentation

12.20.3.1 GetCurveData()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetCurveData (
    AAX_CTypeID iCurveType,
    const float * iValues,
    uint32_t iNumValues,
    float * oValues ) const [pure virtual]
```

Generate a set of output values based on a set of given input values.

This method is used by the host to generate graphical curves. Given a set of input values, e.g. frequencies in Hz, this method should generate a corresponding set of output values, e.g. dB gain at each frequency. The semantics of these input and output values are dictated by `iCurveType`. See [AAX_ECType](#).

Plug-ins may also define custom curve type IDs to use this method internally. For example, the plug-in's GUI could use this method to request curve data in an arbitrary format.

Note

- This method may be called by the host simultaneously from multiple threads with different `iValues`.

Note

- `oValues` must be allocated by caller with the same size as `iValues` (`iNumValues`).

Host Compatibility Notes Versions of S6 software which support the [GetCurveDataDisplayRange\(\)](#) method will not display a plug-in's curve data unless both [GetCurveData\(\)](#) and [GetCurveDataDisplayRange\(\)](#) are supported by the plug-in.

Warning

S6 currently polls this method to update a plug-in's EQ or dynamics curves based on changes to the parameters mapped to the plug-in's EQ or dynamics center section page tables. Parameters that are not included in these page tables will not trigger updates to the curves displayed on S6. (GWSW-7314, [PTSW-195316 / PT-218485](#))

Parameters

in	<i>iCurveType</i>	One of AAX_ECurveType
in	<i>iValues</i>	An array of input values
in	<i>iNumValues</i>	The size of <i>iValues</i>
out	<i>oValues</i>	An array of ouptut values

Returns

This method must return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not support curve data for the requested *iCurveType*

Implemented in [AAX_CEffectParameters](#).

12.20.3.2 GetCurveDataMeterIds()

```
virtual AAX\_Result AAX_IACFEffectParameters_V3::GetCurveDataMeterIds (
    AAX\_CTypeID iCurveType,
    uint32_t * oXMeterId,
    uint32_t * oYMeterId ) const [pure virtual]
```

Indicates which meters correspond to the X and Y axes of the EQ or Dynamics graph.

These meters can be used by attached control surfaces to present an indicator in the same X/Y coordinate plane as the plug-in's curve data.

Parameters

in	<i>iCurveType</i>	One of AAX_ECurveType
out	<i>oXMeterId</i>	Id of the X-axis meter
out	<i>oYMeterId</i>	Id of the Y-axis meter

Returns

This method should return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not implement it.

Implemented in [AAX_CEffectParameters](#).

12.20.3.3 GetCurveDataDisplayRange()

```
virtual AAX\_Result AAX_IACFEffectParameters_V3::GetCurveDataDisplayRange (
    AAX\_CTypeID iCurveType,
    float * oXMin,
    float * oXMax,
    float * oYMin,
    float * oYMax ) const [pure virtual]
```

Determines the range of the graph shown by the plug-in.

Min/max arguments define the range of the axes of the graph.

Parameters

in	<i>iCurveType</i>	One of AAX_ECurveType
out	<i>oXMin</i>	Min value of X-axis range
out	<i>oXMax</i>	Max value of X-axis range
out	<i>oYMin</i>	Min value of Y-axis range
out	<i>oYMax</i>	Max value of Y-axis range

Returns

This method should return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not implement it.

Implemented in [AAX_CEffectParameters](#).

Collaboration diagram for EQ and Dynamics Curve Displays:



12.21 Task agent interface

12.21.1

A mechanism for hosts to request that plug-ins perform tasks.

This interface represents an optional component that you can add to your plug-in in order to support extended features of the AAX SDK.

Host Compatibility Notes This interface is not yet used in any AAX hosts

The plug-in implements an [AAX_ITaskAgent](#), which is used by the host to add or cancel tasks.

The host implements [AAX_IACFTask](#) for task objects representing each task that it wants the plug-in to perform.

To request a task, the host adds a task object to the plug-in's task agent interface.

The type of each task is identified with a four-char ID. If additional arbitrary data is necessary to describe the task it is provided via [data buffers](#).

The plug-in checks this data to understand what work needs to be done, then performs the task. The task may be, and usually is, executed asynchronously. The plug-in optionally updates the task progress as it proceeds, then calls [AAX_ITask::SetDone\(\)](#) when the work is completed. If the task involves returning data back to the host, the plug-in first calls [AAX_ITask::AddResult\(\)](#) one or more times to provide the data via data buffers.

To be available as a task agent, the plug-in's task agent implementation must be registered with the host in the plug-in's Description callback like other process pointers. To add this interface to your plug-in at describe time, call [AAX_IEffectDescriptor::AddProcPtr\(\)](#) using the [kAAX_ProcPtrID_Create_TaskAgent](#) selector.

12.21.2 Communicating with other modules

Like other modules, the task agent interface is provided with a reference to the plug-in's [AAX_IEffectDirectData](#) object at initialization. In order to transfer data between a plug-in's [AAX_IEffectDirectData](#) object and its other objects, dedicated custom data methods in those objects' interfaces should be used. For example, to communicate with the plug-in's data model, use [AAX_IEffectParameters::GetCustomData\(\)](#) and [AAX_IEffectParameters::SetCustomData\(\)](#).

Classes

- class [AAX_CTaskAgent](#)
Default implementation of the [AAX_ITaskAgent](#) interface.
- class [AAX_IACFTask](#)
Versioned interface for an asynchronous task.
- class [AAX_IACFTaskAgent](#)
Versioned interface for a component that accepts task requests.
- class [AAX_ITask](#)
Interface representing a request to perform a task.
- class [AAX_ITaskAgent](#)
Interface for a component that accepts task requests.

Enumerations

- enum class [AAX_TaskCompletionStatus](#) : int32_t {
[AAX_TaskCompletionStatus::None](#) = 0 ,
[AAX_TaskCompletionStatus::Done](#) = 1 ,
[AAX_TaskCompletionStatus::Canceled](#) = 2 ,
[AAX_TaskCompletionStatus::Error](#) = 3 }

12.21.3 Enumeration Type Documentation

12.21.3.1 AAX_TaskCompletionStatus

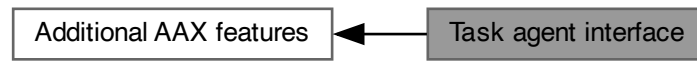
```
enum class AAX\_TaskCompletionStatus : int32_t [strong]
```

Completion status for use with [AAX_ITask::SetDone\(\)](#)

Enumerator

None	
Done	
Canceled	
Error	

Collaboration diagram for Task agent interface:



12.22 AAX Library features

12.22.1

AAX Library core support for the AAX interface

The AAX Library includes several built-in features that are designed to facilitate plug-in development and to make it easy to create plug-ins with correct and consistent behavior. Although these features are not a part of the AAX API, they are a core part of the SDK.

Documents

- [Parameter Manager](#)

Optional (but recommended) system for managing AAX plug-in parameters.

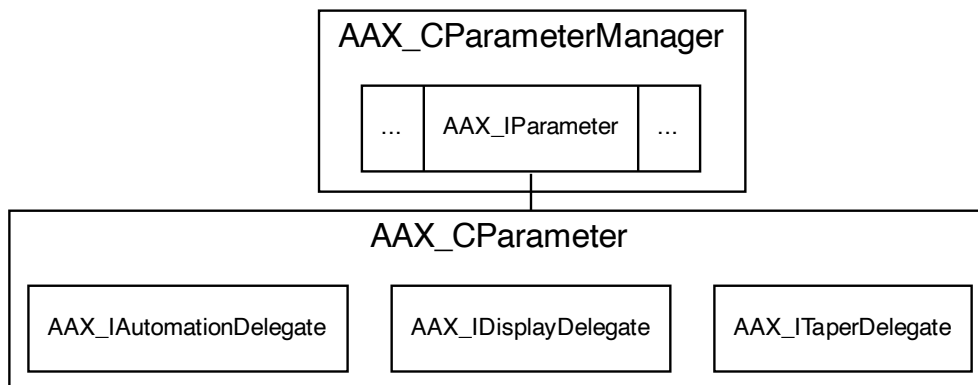
Collaboration diagram for AAX Library features:



12.23 Parameter Manager

12.23.1

Optional (but recommended) system for managing AAX plug-in parameters.



The Parameter Manager is a generic container for a plug-in's parameters, which constitute the complete externally-facing state of a plug-in's data model. Additional internal state data may be stored via settings chunks. The Parameter Manager is owned and operated by the plug-in's [Data model interface](#).

The Parameter Manager provides a convenient and consistent interface by which a plug-in's data model implementation may access its parameters. Other plug-in components that require access to the data model may also use this interface, or a proxy of it, to view the current state of the plug-in.

In the Parameter Manager, implementation-specific parameter behaviors such as taper and display formatting are modular and are applied through delegation. Because of this model, it is possible to easily create a wide variety of behavior combinations without additional subclassing; any display behavior may be combined with any taper behavior, and a newly written behavior can be quickly "mixed in" to many parameters.

12.23.2 Parameter concepts

- [Parameter value domains](#)
- [Taper](#)
- [Delegates](#)
- [Model-View-Controller](#)

12.23.2.1 Parameter value domains

In AAX, parameter values can be represented in one of two "domains". Developers work with parameters in the *real* domain, while the host handles parameters in a scaled, *normalized* format.

Real (or "logical") domain

AAX plug-ins and parameter controllers work with typed parameter values that represent the *real* (logical) state of the parameter. The type, form, and meaning of this value is dependent on the parameter's implementation and is unknown to the host.

Normalized domain

The AAX host works with parameter values that have been scaled (*normalized*) to a type-agnostic format. Although normalized values make little logical sense, they provide the host with a consistent means of handling, storing, and communicating parameters' values without having to worry about the actual implementation or meaning of the parameters. Normalized parameter values are 64-bit floating point and are scaled to a range of [0, 1].

For more information about conversion between parameter domains, see [AAX_IParameter](#).

Note

The [AAX_IEffectParameters](#) interface currently utilizes a secondary normalization to full-scale `int32_t` values. In the future, this will be unified with the double precision floating point normalization documented above.

12.23.2.2 Taper

A *taper* is the conversion function that translates a parameter's value between its real and normalized forms.

For example, a taper could be created that converts between a normalized value ([0, 1]) and a real frequency value ranging from [20 2000]. The conversion between these two ranges could be linear or logarithmic, or could use any other desired mapping. This mapping, as well as the specific range of the possible logical values, is defined by the taper.

For more information about tapers in AAX, see [AAX_ITaperDelegate](#).

12.23.2.3 Delegates

In AAX, individual parameters achieve their own unique behavior by being associated with behavioral delegates.

For example, when [AAX_CParameter::SetNormalizedValue\(\)](#) is called on a particular parameter through its [AAX_IParameter](#) interface, the [AAX_CParameter](#) calls into a [AAX_ITaperDelegate](#) that it owns in order to convert the normalized value to its real equivalent. This real value is then set as the parameter's new state.

For more information about how delegates are used to create a parameter's behavior see [AAX_CParameter](#)

12.23.2.4 Model-View-Controller

AAX adheres roughly to a Model-View-Controller pattern. The Parameter Manager functions within the context of [AAX_IEffectParameters](#), which in turn acts as an AAX plug-in's Data Model in an MVC sense. Views, such as the plug-in's GUI, attached control surfaces, or the automation facilities in the AAX host, are given access to the Data Model via a central Controller, which is represented by the [AAX_IController](#) interface.

For more information about how MVC applies to AAX, see the [Data model interface](#) documentation page.

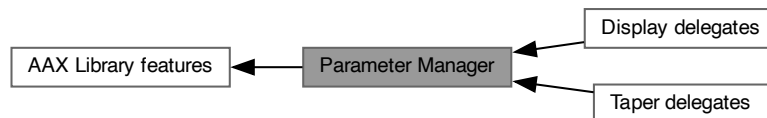
Classes

- class [AAX_CParameter< T >](#)
Generic implementation of an [AAX_IParameter](#).
- class [AAX_CParameterManager](#)
A container object for plug-in parameters.
- class [AAX_IParameter](#)
The base interface for all normalizable plug-in parameters.

Documents

- [Taper delegates](#)
Classes for conversion to and from normalized parameter values.
- [Display delegates](#)
Classes for parameter value string conversion.

Collaboration diagram for Parameter Manager:



12.24 Taper delegates

12.24.1

Classes for conversion to and from normalized parameter values.

Taper delegates are used to convert real parameter values to and from their normalized representations. All taper delegates implement the `AAX_ITaperDelegate<T>` interface template, which contains two conversion functions:

```
virtual T      NormalizedToReal(double normalizedValue) const = 0;
virtual double RealToNormalized(T realValue) const = 0;
```

In addition, tapers may incorporate logical value constraints via the following interface methods:

```
virtual T      GetMaximumValue() const = 0;
virtual T      GetMinimumValue() const = 0;
virtual T      ConstrainRealValue(T value) const = 0;
```

For more information, see the [AAX_ITaperDelegate](#) class documentation.

Classes

- class [AAX_ITaperDelegateBase](#)
Defines the taper conversion behavior for a parameter.
- class [AAX_ITaperDelegate< T >](#)

Collaboration diagram for Taper delegates:



12.25 Display delegates

12.25.1

Classes for parameter value string conversion.

Display delegates are used to convert real parameter values to and from their formatted string representations. All display delegates implement the [AAX_IDisplayDelegate](#) interface, which contains two conversion functions:

```
virtual bool ValueToString(T value, std::string& valueString) const = 0;
virtual bool StringToValue(const std::string& valueString, T& value) const = 0;
```

12.25.2 Display delegate decorators

The AAX SDK utilizes a decorator pattern in order to provide code re-use while accounting for a wide variety of possible parameter display formats. The SDK includes a number of sample display delegate decorator classes.

Each concrete display delegate decorator implements [AAX_IDisplayDelegateDecorator](#) and adheres to the decorator pattern. The decorator pattern allows multiple display behaviors to be composited or wrapped together at run time. For instance it is possible to implement a dBV (dB Volts) decorator, by wrapping an [AAX_CDecibelDisplayDelegateDecorator](#) with an [AAX_CUnitDisplayDelegateDecorator](#).

12.25.2.1 Display delegate decorator implementation

By implementing [AAX_IDisplayDelegateDecorator](#), each concrete display delegate decorator class implements the full [AAX_IDisplayDelegate](#) interface. In addition, it retains a pointer to the [AAX_IDisplayDelegateDecorator](#) that it wraps. When the decorator performs a conversion, it calls into its wrapped class so that the wrapped decorator may apply its own conversion formatting. By repeating this pattern in each decorator, all of the decorator subclasses call into their "wrapper" in turn, resulting in a final string to which all of the decorators' conversions have been applied in sequence.

Here is the relevant implementation from [AAX_IDisplayDelegateDecorator](#) :

```
template <typename T>
AAX_IDisplayDelegateDecorator<T>::AAX_IDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>&
    displayDelegate) :
    AAX_IDisplayDelegate<T>(),
    mWrappedDisplayDelegate(displayDelegate.Clone())
{
}

template <typename T>
bool AAX_IDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString) const
{
    return mWrappedDisplayDelegate->ValueToString(value, valueString);
}

template <typename T>
bool AAX_IDisplayDelegateDecorator<T>::StringToValue(const AAX_CString& valueString, T* value) const
{
    return mWrappedDisplayDelegate->StringToValue(valueString, value);
}
```

12.25.2.2 Decibel decorator example

Here is a concrete example of how a decibel decorator might be implemented

```
template <typename T>
bool    AAX_CDecibelDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString) const
{
    if (value <= 0)
    {
        *valueString = AAX_CString("--- dB");
        return true;
    }

    value = 20*log10(value);
    bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
    *valueString += AAX_CString("dB");
    return succeeded;
}
```

Notice in this example that the [ValueToString\(\)](#) method is called in the parent class, [AAX_IDisplayDelegateDecorator](#). This results in a call into the wrapped class' implementation of [ValueToString\(\)](#), which converts the decorated value to a redecorated string, and so forth for additional decorators.

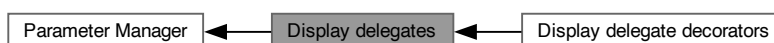
Classes

- class [AAX_IDisplayDelegateBase](#)
Defines the display behavior for a parameter.
- class [AAX_IDisplayDelegate< T >](#)

Documents

- [Display delegate decorators](#)
Classes for adapting parameter value strings.

Collaboration diagram for Display delegates:



12.26 Display delegate decorators

12.26.1

Classes for adapting parameter value strings.

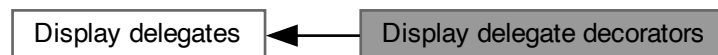
The AAX parameter display strategy uses a decorator pattern for parameter value formatting. This approach allows developers to maximize code re-use across display delegates with many different kinds of varying formatting, all without creating interdependencies between the different display delegates themselves.

For more information, see [Display delegate decorators](#). For even more information, about the Decorator design pattern, please consult the GOF design patterns book.

Classes

- class [AAX_CDecibelDisplayDelegateDecorator< T >](#)
A percent decorator conforming to [AAX_IDisplayDelegateDecorator](#).
- class [AAX_CPercentDisplayDelegateDecorator< T >](#)
A percent decorator conforming to [AAX_IDisplayDelegateDecorator](#).
- class [AAX_CUnitDisplayDelegateDecorator< T >](#)
A unit type decorator conforming to [AAX_IDisplayDelegateDecorator](#).
- class [AAX_CUnitPrefixDisplayDelegateDecorator< T >](#)
A unit prefix decorator conforming to [AAX_IDisplayDelegateDecorator](#).
- class [AAX_IDisplayDelegateDecorator< T >](#)
The base class for all concrete display delegate decorators.

Collaboration diagram for Display delegate decorators:



12.27 Additional Topics

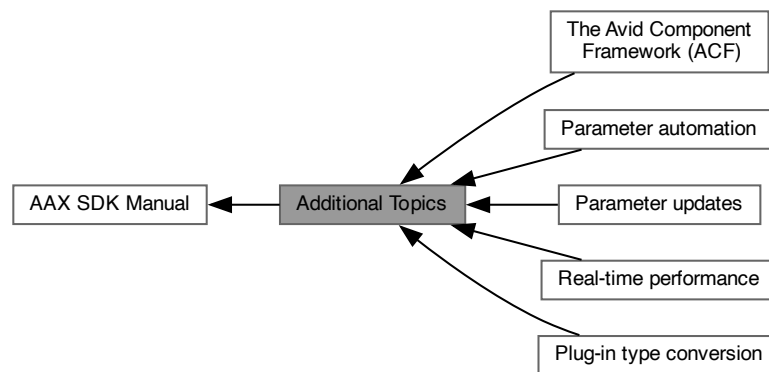
12.27.1

Additional information about the AAX design.

Documents

- [Real-time performance](#)
Guidelines for avoiding audio streaming errors.
- [Parameter automation](#)
Information about parameter automation.
- [Parameter updates](#)
The anatomy of a parameter update.
- [Plug-in type conversion](#)
Specification for valid conversions between plug-in types.
- [The Avid Component Framework \(ACF\)](#)
How the AAX C++ interfaces work.

Collaboration diagram for Additional Topics:



12.28 Real-time performance

Guidelines for avoiding audio streaming errors.

This page provides an overview of best practices for avoiding streaming errors and achieving good performance for audio processing on real-time threads.

These recommendations are based on observations we have made when reviewing common Pro Tools streaming errors, especially those caused by plug-ins, as well as on information we have gathered from a number of partners and other experts in the field.

See also

[Plug-In Causes Audio Streaming Errors](#)

12.28.1 Things NOT To Do In An Audio Plug-In Render Callback

- No unbounded calls/loops
- No access to paged memory or files
- No system calls
- No memory allocations or deallocations
- No exceptions
- No locks (priority inversions)
- No data races
- Avoid context switches
- No Objective-C or Swift code. These can incur system calls and take locks - see this article for more information.
- Do not use the JUCE `callAsync()` function - it is not real-time safe. As an alternative, you can use a separate dedicated thread that wakes on a timer or the [AAX TimerWakeup\(\)](#) method to handle non-real time work.
- Never perform PACE license checks in audio processing thread; always add code annotations to prevent license checks in any code which will be executed from the audio processing callback.

12.28.2 Things To Do In An Audio Plug-In Render Callback

- If passing data, always use lock-free FIFOs
 - When making data from other parts of the plug-in available to its real-time callback you should always use the [AAX](#) packet system; this will ensure thread safety, proper timing of the data delivery with respect to the audio being processed, and optimal real-time thread performance.
 - There is a nice reference implementation for a general-purpose FIFO in the [farbot](#) project
 - Lock-free FIFOs are also good for passing data from the real-time thread to a low-priority thread in order to do heavy lifting like writes to disk
- If sharing small amounts of data (≤ 8 bytes), use atomics
 - Make a local copy/cache of any atomic values that need to be read multiple times from your render function
 - When using atomics, always make the compiler prove that its implementation is lock-free e.g. using a `static_assert` that `std::atomic<T>::is_lock_free`
- When sharing larger data, if it is acceptable if the data sometimes cannot be accessed, use a `try_lock()` in the real-time thread and a `lock()` in any non-real-time threads.
 - When using this strategy you should use a spin lock, not a `std::mutex`; `std::mutex::unlock()` can block in a system call to wake the waiting thread
- When sharing larger data, if it is acceptable to access a stale copy of the data, then use a compare-and-exchange loop
 - Be careful about memory leaks with this strategy
 - See the `NonRealtimeMutable` template in the [farbot](#) project

12.28.3 Good Resources And Examples

- [Real-time audio programming 101: time waits for nothing](#) by Ross Bencina
- [Four common mistakes in audio development](#) by Michael Tyson
- [farbot: FAbian's Realtime Box o' Tricks](#) project on GitHub

Collaboration diagram for Real-time performance:



12.29 Parameter automation

Information about parameter automation.

12.29.1 On this page

12.29.2 Overview

The term "automation" can mean two things in AAX:

1. A host feature allowing users to record and play back plug-in parameter changes. In this documentation, this data is referred to as **automation data**, and it is stored in **automation lanes** in the host.
2. A system for arbitrating between changes from different parameter editors such as the plug-in GUI, control surfaces, and pre-recorded automation values. In this documentation, this is referred to as the **event system** for parameters.

Here are some examples of how these two different meanings are used in AAX:

- The [AAX_IAutomationDelegate](#) provides methods for interacting with the host's parameter event system.
- [AAX_IACFEffParameters::GetParametersIsAutomatable\(\)](#) and the `automatable` parameter in the [AAX_CParameter](#) constructor reflect whether a parameter can have automation written and read by the host.
- [AAX_IController::GetCurrentAutomationTimestamp\(\)](#) gets the timestamp for pre-recorded automation data when it is received by the plug-in during playback

For more information about the parameter event system, see the [Parameter updates](#) pages, and particularly the information on the [Token protocol](#)

12.29.3 Plug-in elements used for automation

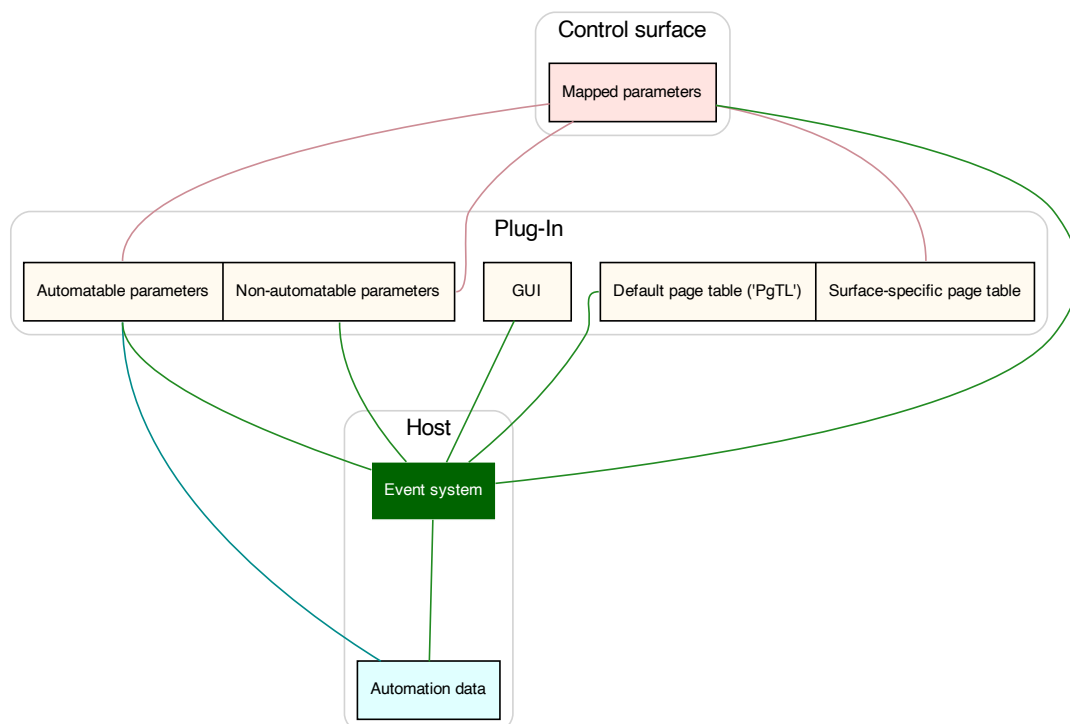


Figure 12.4 Plug-in elements used for events and automation

12.29.3.1 Defining automatable parameters

In order for a parameter to be available for automation recording, editing, and playback, the plug-in must meet the following criteria:

- It must provide `true` when the host calls [GetParameterIsAutomatable\(\)](#) for the parameter. In nearly all plug-ins, this means providing `true` to the `automatable` parameter in the parameter's [AAX_CParameter](#) constructor.
- It must expose the parameter to the parameter event system (see below.)

In order for a parameter to be exposed to the event system, the plug-in must meet the following criteria:

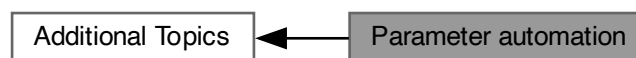
- It must respond to all parameter methods in the [AAX_IEffectParameters](#) interface, particularly [GetNumberOfParameters\(\)](#) and [GetParameterIDFromIndex\(\)](#). Generally this is accomplished by adding an [AAX_CParameter](#) object for each parameter to the plug-in's [Parameter Manager](#).
- It must include the parameter in its one-parameter-per-page 'PgTL' (default) page tables. See [Implementing Page Tables](#) in the [Page Table Guide](#) for more information about defining this page table type.

All plug-in parameters must be registered with the host's event system in order for editors, including the plug-in's GUI, to work properly. Therefore a plug-in should always define a complete 'PgTL' (default) page table including all of its parameters, even the parameters that are not "automatable".

12.29.4 Advanced automation topics

- [Linked parameters](#)

Collaboration diagram for Parameter automation:



12.30 Parameter updates

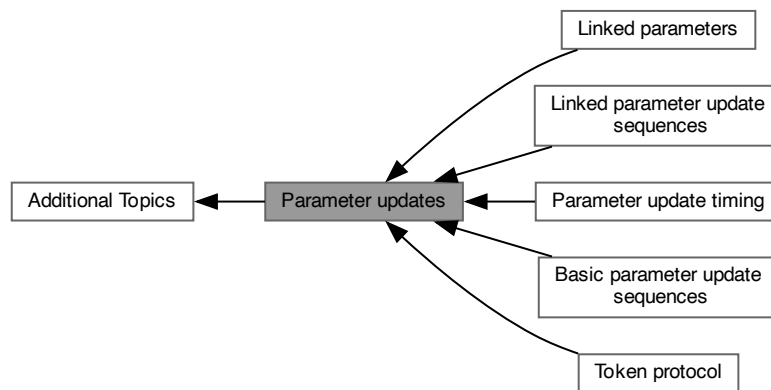
12.30.1

The anatomy of a parameter update.

Documents

- [Parameter update timing](#)
Details about parameter timing and how to keep parameter updates in sync.
- [Token protocol](#)
Communicating parameter state with the host.
- [Basic parameter update sequences](#)
Sequence diagrams for some common parameter update scenarios.
- [Linked parameters](#)
How to link parameters.
- [Linked parameter update sequences](#)
Sequence diagrams for some common linked parameter update scenarios.

Collaboration diagram for Parameter updates:



12.31 Parameter update timing

Details about parameter timing and how to keep parameter updates in sync.

12.31.1 On this page

- [Timeline Locations](#)
- [Coordinating the data model and algorithm](#)
- [Fixing timing issues due to shared data](#)
- [Determining the absolute timestamp for a parameter update](#)

12.31.2 Timeline Locations

At any given moment, a plug-in may be asked to handle events from multiple locations on the timeline. Each module in an AAX plug-in may be updated using a different timeline position. For example:

- During automation playback the host may choose to send parameter updates in advance, while the algorithm is still processing audio from earlier in the timeline.
- When a processing chain involves a significant amount of latency, the host may delay the metering data which is available to the plug-in's GUI until the point in time when the corresponding processed audio is actually being played back to the user.

In this article, we will refer to the following timeline locations:

- *Automation time*: The location that corresponds to the state of the plug-in's data model
- *Playhead*: The location where the audio engine is currently gathering samples for processing
- *Render time*: The location of the audio samples currently being processed by the plug-in's algorithm
- *Presentation time*: The location that corresponds to the playback presentation to the user (i.e. the sound coming out of the speakers)

Figure 1: Timeline locations

12.31.3 Coordinating the data model and algorithm

As an AAX plug-in developer, you don't usually need to worry about the fact that your plug-in's data model and algorithm may each represent a different point in the timeline; the [AAX packet system](#) handles all of the necessary synchronization between these two locations.

This works seamlessly in a normal AAX plug-in because the real-time algorithm is fully decoupled from the plug-in's data model. Since all of the state information for the algorithm is delivered through its [context structure](#), the host can simply swap in the correct context data for each call to the processing callback. The plug-in does not require any special handling code to synchronize between the two timeline locations, and, as a bonus, AAX plug-ins can achieve deterministic, accurate automation playback without doing any extra work to handle time-stamped parameter update queues or other overhead.

Figure 2: Synchronization through the AAX packet system

12.31.3.1 A closer look at the AAX packet delivery system

Adding new packets for automation events

When playing back automation, the AAX host calls [UpdateParameterNormalizedValue\(\)](#) to update the data model state, then calls [GenerateCoefficients\(\)](#) to trigger the generation of new packets. See [Basic parameter update sequences](#) for a full description of this sequence.

Before the host calls [GenerateCoefficients\(\)](#) to generate packets for an automation breakpoint, it records the timeline position of the breakpoint ([AAX_IController::GetCurrentAutomationTimestamp\(\)](#) provides this value as a sample offset from the beginning of playback.) Every packet that is posted during execution of [GenerateCoefficients\(\)](#) is tagged with this timestamp when it is queued for delivery.

Packet delivery for AAX Native plug-ins

As the playhead advances and sample buffers are queued for processing, the host tracks the location of the next time-stamped packet in the packet queue. As the render time location for a Native plug-in processing chain approaches the next packet time-stamp for a plug-in in the chain, the host divides the plug-in's processing buffers into smaller buffers. When the render time location is as close as possible to the packet's time-stamp, the host delivers the packet. The packet data is available to the algorithm in its context the next time it is executed.

Because the host may divide native processing buffers down to a minimum size of [AAX_eAudioBufferLengthNative_Min](#) - 32 samples - the host can guarantee that all automation playback will be effected within 32 samples of the actual automation breakpoint location. In addition, with the help of some extra internal bookkeeping, AAX hosts also guarantee that the exact sample where an automation breakpoint is applied will be deterministic and will not change between different playback passes.

Packet delivery for AAX DSP plug-ins

The packet delivery system for AAX DSP plug-ins works similarly to the system for AAX Native plug-ins. AAX DSP plug-ins use a fixed buffer size, so the host is not able to divide their playback buffers into smaller units: the plug-in will receive each data packet in the fixed-size playback buffer which most closely corresponds to the location of the automation event which triggered the packet.

An AAX DSP plug-in which declares an [AAX_eProperty_DSP_AudioBufferLength](#) value of N will be guaranteed to receive data packets within N/2 samples of the actual automation event position on the timeline. Since the default buffer size for an AAX DSP plug-in is 4 samples, this yields extremely accurate automation playback with no extra work required in the plug-in algorithm.

12.31.4 Fixing timing issues due to shared data

The packet system works perfectly to synchronize the states of the plug-in data model and algorithm, *but only when the plug-in algorithm is fully decoupled from the data model*. If the algorithm directly shares data with the data model then the algorithm will immediately start using any new data model state without waiting for the corresponding coefficient delivery.

Figure 3 shows one kind of problem that can arise when a plug-in uses the same state for both its data model and its algorithm. In this case, the plug-in applied a volume trim (shown in the automation lane at the top of the image) to its algorithm as soon as the parameter update was applied to its data model, even though the algorithm was not yet processing the audio at the Automation time location. As a result, the audio trim was applied several hundred samples too early.

Figure 3: Offset automation playback due to lack of timeline location synchronization in a monolithic plug-in

12.31.4.1 Monolithic plug-ins

Plug-ins that share data directly between their data model and algorithm are referred to as *monolithic*. All plug-ins that inherit from the [AAX_CMonolithicParameters](#) helper class are monolithic.

Note

Monolithic plug-ins must always set the [AAX_eProperty_Constraint_Location](#) property to include [AAX_eConstraintLocationMask_DataModel](#) in order to avoid being loaded into incompatible AAX hosts.

All monolithic plug-ins must include special handling code to reconcile the plug-in's automation time state with its render time state.

12.31.4.2 How to resolve timing errors

There are many possible solutions for the timing errors that arise when a plug-in combines data from different time locations. Ultimately, the plug-in must separate the state that is represented at different time locations.

In most cases, this requires deferring data model state changes from being applied to the algorithm until the relevant samples are being processed in the render callback. One easy way to accomplish this separation is to take advantage of the synchronization provided by the AAX packet delivery system. This approach benefits from the fact that it emulates the design of a normal, decoupled AAX plug-in.

After a packet is queued with a call to `PostPacket()`, the packet delivery system will wait to update the algorithm's context structure with the packet's data until the Render time location is very close to the automation event (see [above](#).) This provides an appropriate mechanism for deferring state changes in the plug-in's data model until the Render time location has "caught up" to the correct sample.

Figure 4 shows the same scenario as Figure 3, but now the plug-in has been updated to defer data model updates from the automation time location so that they are applied as coefficients in the algorithm when the render time location has reached the correct point on the timeline.

Figure 4: Deferring a data model update in a monolithic plug-in using the packet queue

Here is one way to use the packet delivery system to defer changes to the data model state:

```
AAX_Result
MyEffectParameters::UpdateParameterNormalizedValue(
    AAX_CParamID iParamID,
    double aValue,
    AAX_EUpdateSource inSource)
{
    // Call inherited
    AAX_Result result = AAX_CMonolithicParameters::UpdateParameterNormalizedValue(
        iParamID,
        aValue,
        inSource);
    if (AAX_SUCCESS != result) { return result; }

    // Do whatever additional work is required to note that the
    // parameter has been updated - for example, set a "dirty"
    // flag for the parameter.

    return result;
}

AAX_Result
MyEffectParameters::GenerateCoefficients()
{
    // Call inherited
    AAX_Result result = AAX_CMonolithicParameters::GenerateCoefficients();
    if (AAX_SUCCESS != result) { return result; }

    const uint32_t stateNum = mMyStateCounter++; // member uint32_t

    // Do whatever additional work is required to capture the current
    // parameter state and associate it with stateNum, for example
    // check for "dirty" parameters and create a list of these
    // parameters with their values, add this list to a map using
    // stateNum as a key, and clear the "dirty" flags.

    result = Controller()->PostPacket(
        kCurrentStateFieldIndex,
        &stateNum,
        sizeof(uint32_t));

    return result;
}

struct MyContextStructure
{
    int32_t * mCurrentStateNum; // Private data
    // ...
};

void
MyAudioRenderCallback(
    MyContextStructure* const inInstancesBegin [],
    const void* inInstancesEnd)
{
    /* For each instance... */
    const uint32_t stateNum = instance->mCurrentStateNum;
```

```

// Update the custom plug-in object state based on stateNum
// and the additional data that was cached during
// GenerateCoefficients().
}

```

Figure 5: One specific solution for deferring a data model update in a monolithic plug-in using the packet queue

This approach is incorporated directly into the design of [AAX_CMonolithicParameters](#). If your plug-in data model is a subclass of [AAX_CMonolithicParameters](#) then you can follow these steps to ensure accurate parameter update timing in your plug-in:

1. After creating an automatable parameter, call [AAX_CMonolithicParameters::AddSynchronizedParameter\(\)](#) to add the parameter to an internal list of parameters to synchronize using the deferred-update system
2. In the plug-in's [RenderAudio\(\)](#) implementation, iterate through the incoming queue of deferred parameter values
3. Update the coefficients used by the plug-in's algorithm or other processing components

NOTES

- Remember to use the deferred parameter values, not values of the plug-in's [AAX_IParameter](#) objects, when setting the state of the plug-in's coefficients
- The deferred parameter values are delivered in the real-time thread, so all synchronized updates should follow the basic principles of real-time operation such as avoiding memory allocation/free, thread synchronization, access to shared resources, or any other actions which could block the real-time thread

For reference, see [DemoMIDI_Synth](#) and the other example instrument plug-ins. All of the instrument examples in the AAX SDK use these facilities to achieve deterministic, accurate playback for automated parameters.

One benefit of this approach is that it provides a compatible interface with monolithic plug-in objects which are designed to work across multiple plug-in formats. For example, the set of parameter updates provided to [AAX_CMonolithicParameters::RenderAudio\(\)](#) "RenderAudio" can be provided to plug-in objects which require a queue of time-stamped parameter updates for each audio render callback.

12.31.4.3 Additional considerations

Of course, the approach described in this section is just one possible solution. The [timestamp](#) section below provides some alternatives to using the packet queue system for synchronization. Ultimately, the best design for your plug-in will depend on the facilities that are available in the plug-in's monolithic state object, the size of this object, its interface, the number of parameters representing its state, and other internal details.

Here are some additional factors to consider when using the packet queue system for time location synchronization of parameter updates:

- The algorithm callback / [RenderAudio\(\)](#) method is called from a real-time thread, and may be called concurrently with data model methods. You should use a synchronization strategy that is optimized for high performance in this thread.
- If a parameter is not automatable then you should probably ignore these additional steps and directly update the plug-in's monolithic state object from within [UpdateParameterNormalizedValue\(\)](#) when that parameter is changed. Updates for non-automatable parameters can always be applied to the algorithm "as soon as possible".
- Depending on your plug-in's design you may not need or want to apply this solution to some automatable parameters either. For example, parameters that are unlikely to be automated or which require CPU-intensive changes in your instrument object should probably be updated on the object directly from within [UpdateParameterNormalizedValue\(\)](#), and not from within the real-time thread

12.31.5 Determining the absolute timestamp for a parameter update

The AAX packet queue provides a host-managed system for applying parameter updates at the correct location without requiring any special knowledge about the timeline. However, In some situations a plug-in may need to know the absolute sample position of a parameter change.

For example, a plug-in that synchronizes parameter changes to some external system, and which wants to forward these changes over to the external system as early as possible, would want to know the sample position for a coefficient update when the update is first triggered by a call to [GenerateCoefficients](#).

In these situations it is not suitable to simply use a method like [AAX_ITransport::GetCurrentNativeSampleLocation\(\)](#) which returns the current position of the audio render thread. The parameter update may be occurring at a different location on the timeline from the current render position, so using the current render position for the update would result in timeline offset problems similar to those described above.

12.31.5.1 Obtaining timeline information

AAX provides a variety of information that can be used for timeline synchronization. This information is provided through a combination of [AAX_ITransport](#), [AAX_IController](#), and MIDI beat clock data. Here is a summary of the relevant ways that a plug-in can get information about the timeline and timing synchronization data:

- [AAX_ITransport::GetCurrentNativeSampleLocation\(\)](#) Provides the absolute sample position of the first sample in the audio buffer that is currently being processed by the plug-in's worker chain
- [AAX_IController::GetTODLocation\(\)](#) Provides the current "time of day" value, which is a counter within the audio engine that counts the number of samples that the playhead has traversed since playback start
- [AAX_IController::GetCurrentAutomationTimestamp\(\)](#) Must be called from within [GenerateCoefficients](#). Provides the timestamp for the beginning of the hardware audio buffer during which the generated coefficients will be applied to the algorithm. This timestamp is provided in terms of the "time of day" counter, i.e. the number of samples since playback started.
- MIDI Beat Clock Sends transport start/continue/stop events to plug-ins that register global MIDI nodes

12.31.5.2 Determining the timeline position of a parameter update

Each of the available methods for getting information about the timeline position has a particular purpose. No single interface method can be used to directly determine the sample location for a parameter update, but it is possible to determine this value by combining information from a few of the available methods.

Here are some possible approaches for determining the timeline position of a parameter update

Note

Remember that these are not strict recipes; the specific requirements for what kinds of timeline information are needed will vary from plug-in to plug-in. You may be able to refine these approach to better match the needs of your specific plug-in.

12.31.5.2.1 1. Defer the update to the real-time thread

1. Queue state updates using a plug-in design similar to the one described [above](#)
2. When a state update is received on the real-time thread, call [GetCurrentNativeSampleLocation](#) to get the sample location for the start of the current render buffer
3. Perform all necessary update handling using this value as the sample location

NOTES

- This approach yields a sample location value which is accurate within 32 samples
- Event handling must be performed on the real-time render thread, which may not be viable depending on the types of operations that the plug-in must perform
- Event handling cannot be performed in advance to reduce overall system latency

12.31.5.2.2 2. Compute the timestamp as a TOD offset

1. Add a queue for update events which will be used internally within the plug-in's [AAX_IEffectParameters](#) object
2. In [UpdateParameterNormalizedValue](#), enqueue an update event
3. In [GenerateCoefficients](#), call [AAX_IController::GetTODLocation\(\)](#) and [AAX_IController::GetCurrentAutomationTimestamp\(\)](#)
4. Subtract the current TOD value from the automation timestamp to find the number of samples currently lie between the data model location and the render audio location on the timeline
5. Call [AAX_ITransport::GetCurrentNativeSampleLocation\(\)](#) and add the resulting value to the sample offset that was determined in the last step. The sum of these two values is the approximate absolute sample location for the coefficient update.
6. Once this sample location has been calculated, dequeue all pending update events and handle them using the calculated timestamp

The reason that this approach yields an approximate value is that the TOD location and current playback location are both given in terms of the real-time audio workers, and these values continue to progress simultaneously with execution of methods on the automation update thread. As a result, this approach will yield an absolute timestamp that is "late" by between zero and one hardware buffer.

NOTES

- Using this approach it is possible to handle parameter updates in advance to reduce overall system latency
- This approach yields a sample location value which is accurate within one hardware buffer
- This approach uses AAX interface methods that are not supported in older AAX hosts such as Pro Tools 10

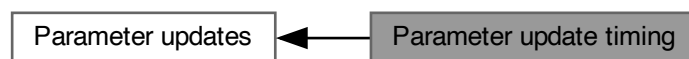
12.31.5.2.3 3. Compute the timestamp with improved accuracy using MIDI Beat Clock You can refine the approach described above by using MBC events to detect the location of playback start.

1. Register a global MIDI node in your plug-in using [AAX_IEffectDescriptor::AddControlMIDINode\(\)](#) with [AAX_eMIDINodeType_Global](#) and the appropriate event mask bitfield for MBC events
2. Override [AAX_IEffectParameters::UpdateControlMIDINodes\(\)](#) to receive MBC data
3. When an MBC Start or Continue event is received, call [AAX_ITransport::GetCurrentNativeSampleLocation\(\)](#) to get the current render location. Cache this value. This value should represent the absolute playback start sample since audio render will not have started before the MBC event dispatch.
4. As in the previous solution, queue relevant update events in [UpdateParameterNormalizedValue](#)
5. In [GenerateCoefficients](#), call [GetCurrentAutomationTimestamp](#) and add the resulting value to the cached playback start sample location
6. Dequeue all pending update events and handle them using the calculated absolute sample timestamp

NOTES

- Using this approach it is possible to handle parameter updates in advance to reduce overall system latency
- This approach will yield timestamps within a few samples of the actual automation event location on the Pro Tools timeline
- This approach uses AAX interface methods that are not supported in older AAX hosts such as Pro Tools 10

Collaboration diagram for Parameter update timing:



12.32 Token protocol

Communicating parameter state with the host.

12.32.1 On this page

- [An Introduction to Tokens](#)
- [Basic Token Operation](#)

12.32.2 An Introduction to Tokens

One way in which a plug-in can communicate with the "outside world" is through Shared Data Services, also known as the Token System. This is a mechanism that allows Pro Tools to share parameter information with external hardware and software modules. While the AAX SDK only uses the Token System indirectly, knowing how it works will provide a good understanding of how linked parameters should operate.

12.32.2.1 Touch

Touch tokens inform the system of user interaction with a parameter. When a parameter is being touched the system knows to stop sending automation data to the plug-in and just use the SET value of the parameter. It is also used to tell the system when to start/stop recording new automation data.

In AAX, the touch message is sent to the host by [AAX_IAutomationDelegate::PostTouchRequest\(\)](#). The most common way to call this method is via the following methods:

```
class AAX_IEffectParameters
{
    virtual AAX_Result TouchParameter ( AAX_CParamID inParameterID );
    virtual AAX_Result ReleaseParameter ( AAX_CParamID inParameterID );
};

class AAX_IParameter
{
    virtual void Touch ();
    virtual void Release ();
};
```

However, AAX plug-ins will rarely need to call these methods directly since the [AAX_CParameter](#) and [AAX_CEffectParameters](#) implementations will automatically handle parameter touch and release tokens whenever a new value is set on the parameter by the plug-in.

Other clients besides the plug-in may touch a parameter. Since the TOUCH token can come from a control surface the touch state will actually come back to the plug-in via:

```
class AAX_IEffectParameters
{
    virtual AAX_Result UpdateParameterTouch ( AAX_CParamID iParameterID, AAX_CBoolean iTouchState );
};
```

This method is mainly important for [linked parameters](#).

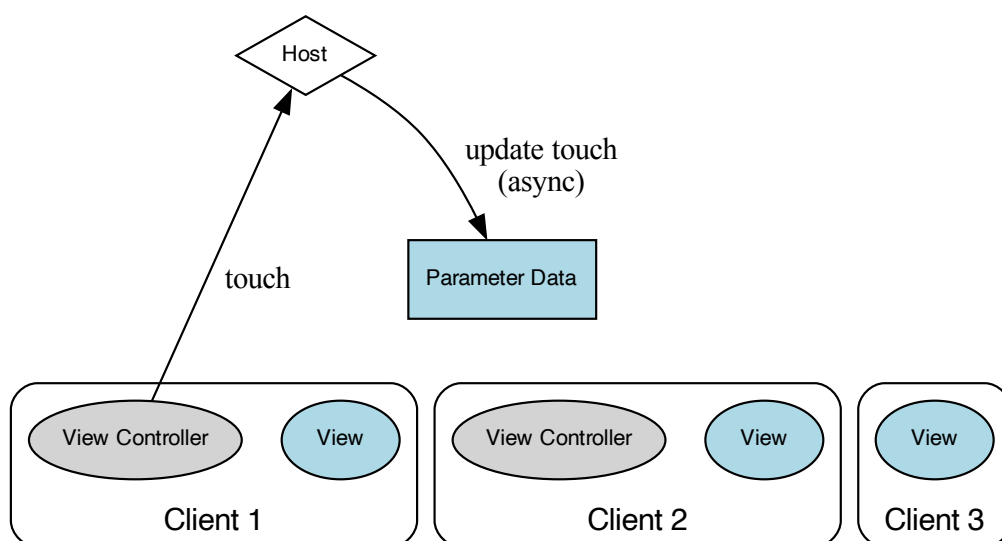


Figure 12.5 Touch request from a view controller, with resulting async touch update

12.32.2.2 Set

SET tokens can come from many different locations: the plug-in GUI, a control surface, loading a chunk or automation playback. Eventually the value of a SET token comes into the plug-in and that's when the internal value of the parameter gets updated. In AAX the SET token will be sent as a result of calling the following method:

```
class AAX_CParameter<T>
{
    void SetValue ( T newValue );
};
```

which will be called from many other supporting methods:

```
class AAX_CParameter<T>
{
    bool SetValueWithBool ( bool value );
    bool SetValueWithInt32 ( int32_t value );
    bool SetValueWithFloat ( float value );
    bool SetValueWithDouble ( double value );
    void SetToDefaultValue ();
    void SetNormalizedValue ( double normalizedNewValue );
    bool SetValueFromString ( const AAX_CString & newValueString );
};
```

When a SET token enters the system from the GUI, control surface or automation the value comes bak to the plug-in via the following method:

```
class AAX_CEffectParameters
{
    AAX_Result UpdateParameterNormalizedValue ( AAX_CParamID iParamID, double aValue, AAX_EUpdateSource inSource );
};
```

At this point the internal contents of the plug-in are set.

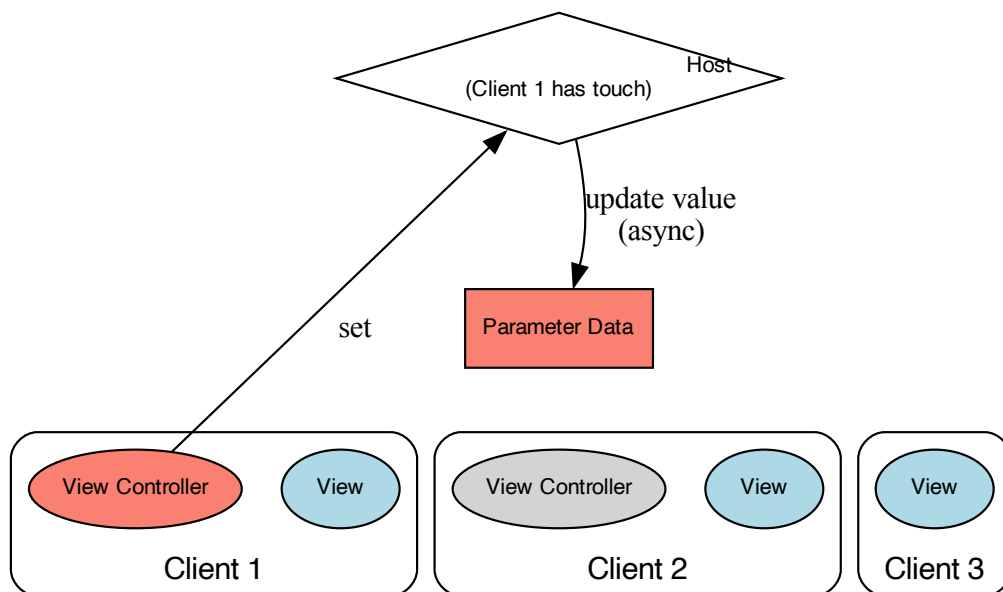


Figure 12.6 Set token asynchronously changes state of the parameter data

12.32.2.3 Update

An update token is generated when the internal value of a parameter has been set. GUIs and control surfaces listen for UPDATE tokens to update the displayed values. In AAX the UPDATE token is sent by calling the following method:

```
class AAX_CParameter<T>
{
    void UpdateNormalizedValue ( double newNormalizedValue );
};
```

All views of the parameter are then asynchronously notified that the value has changed. The plug-in GUI is notified via a call to `AAX_IEffectGUI::ParameterUpdated()`.

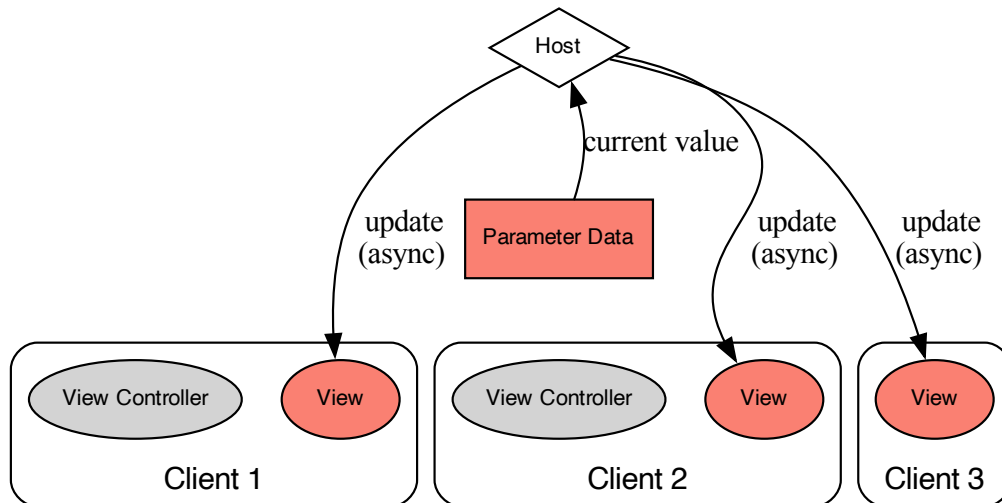


Figure 12.7 Update token triggers async updates to all views

12.32.3 Basic Token Operation

The lists below indicate how the system works in a few different standard update scenarios. To enable logging for these events set `DTF_AUTOMATION=file@DTP_LOW` in the [DigiTrace](#) configuration file. For more detailed information about the sequence of calls used to update parameters in different situations, see [Basic parameter update sequences](#).

12.32.3.1 User Editing

1. User clicks on a parameter in the GUI or grabs a parameter on the controls surface. A TOUCH token should be sent at this point.
2. The user changes the parameter from the GUI or controls surface. A SET token should be sent at this point.
3. The SET token goes into the system and comes back to the plugin via `UpdateParameterNormalizedValue()`.
4. The plug-in updates it's internal state and sends an UPDATE token.
5. Repeat steps 2-4 while changing the parameter.
6. The user lets go of the GUI or controls surface. A TOUCH token with the released state should be sent.

12.32.3.2 Automation Playback

1. The SET token comes from the automation system and enters the plugin via `UpdateParameterNormalizedValue()`.
2. The plug-in updates it's internal state and sends an UPDATE token.
3. Repeat steps 1-2 while playing back automation.

12.32.3.3 Chunk Restoring

1. Plug-in loads the chunk.
2. The plug-in sets every parameters value. Another thing to note is that the
3. `SetValue()` method also contains `Touch()` and `Release()` calls. So, while setting every parameter there is a combination of TOUCH and SET tokens sent to the system.
4. The SET tokens comes back to the plugin via `UpdateParameterNormalizedValue()`.
5. The plug-in updates it's internal state and sends out UPDATE tokens.

Collaboration diagram for Token protocol:



12.33 Basic parameter update sequences

Sequence diagrams for some common parameter update scenarios.

12.33.1 On this page

- [User-generated update](#)
- [Automation playback](#)
- [Initialization](#)

Note

To enable logging for these events at run time set `DTF_AUTOMATION=file@DTP_LOW` in the [DigiTrace](#) configuration file.

12.33.1.1 Notes on threading for these sequences

- Calls from the host into [AAX_IEffectParameters](#) may occur on any thread. In general, the only synchronization that is guaranteed for data model calls in these diagrams is that the call will follow whatever event is indicated as its trigger.
- Calls from the host into [AAX_IEffectGUI](#) will occur on the main application thread unless indicated otherwise in the [AAX_IEffectGUI](#) documentation.
- Host-driven updates to the algorithm context are always synchronized with the real-time processing thread

12.33.2 User-generated update

This is the sequence of calls for a basic, unlinked parameter update triggered by the user. For this sequence, we assume that the edit was triggered by a GUI event.

12.33.2.1 High-level interface calls and events



Figure 12.8 High-level sequence of interface calls and events for a parameter update following a user-generated edit

12.33.2.2 Detailed sequence for default implementation

Note that this diagram assumes a GUI implementation that uses [SetParameterNormalizedValue\(\)](#). The implementation could also use other parameter set methods, either in [AAX_IEffectParameters](#) or directly on an [AAX_IParameter](#). The overall sequence would remain the same.

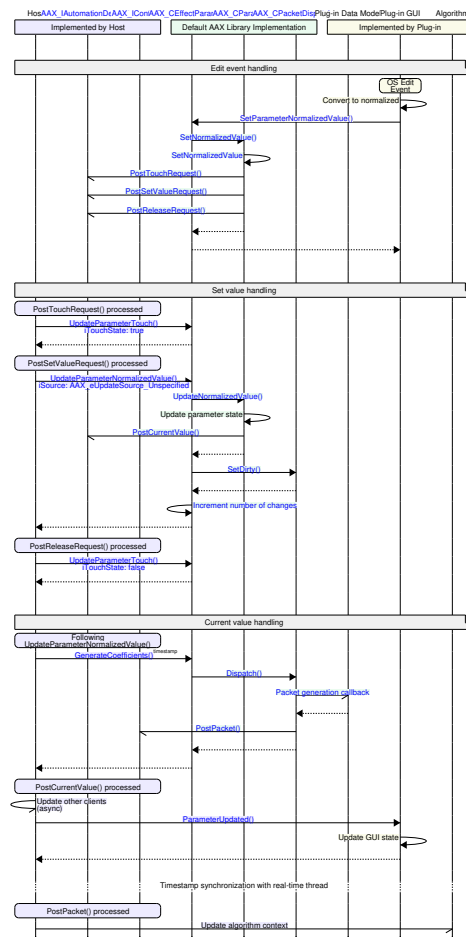
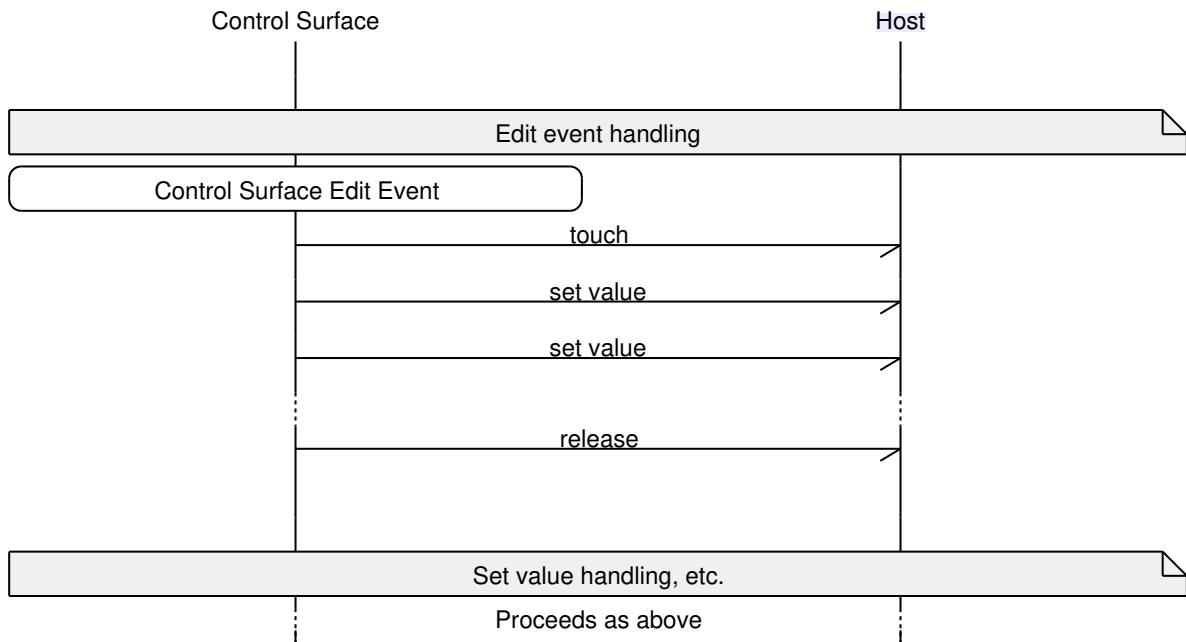


Figure 12.9 Detailed sequence of method calls and events for a parameter update following a user-generated edit on the plug-in GUI

12.33.2.3 Updates from control surfaces

Updates from control surfaces are handled in exactly the same way. In this case, though, the parameter touch, set value, and release tokens are generated by the control surface.



12.33.3 Automation playback

Automation playback handling is similar to the handling for user-generated parameter updates. However, parameters are never touched/released during automation playback. This allows touches from other clients, such as the GUI or control surfaces, to override the automation playback.

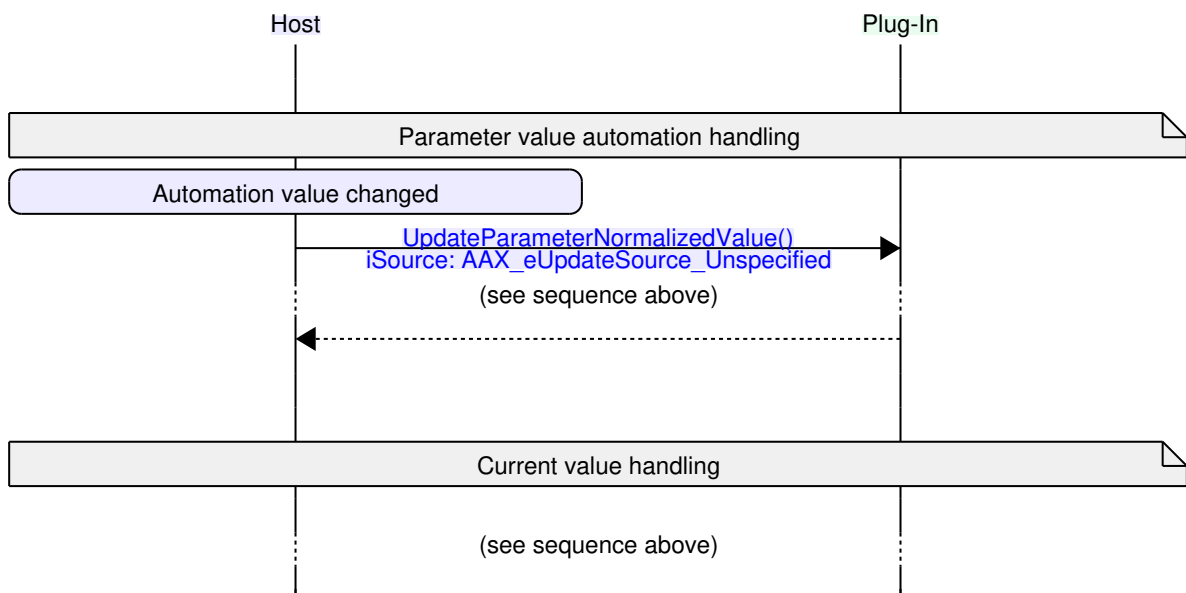


Figure 12.10 Sequence of method calls and events for playback of parameter automation

12.33.4 Initialization

This is the sequence of calls for the initial parameter updates made during data model initialization. Steps that are redundant with sections of the [standard user-generated update sequence](#) are elided.

Todo Update this section with information about default chunk setting, which is a separate step following the procedure described below.

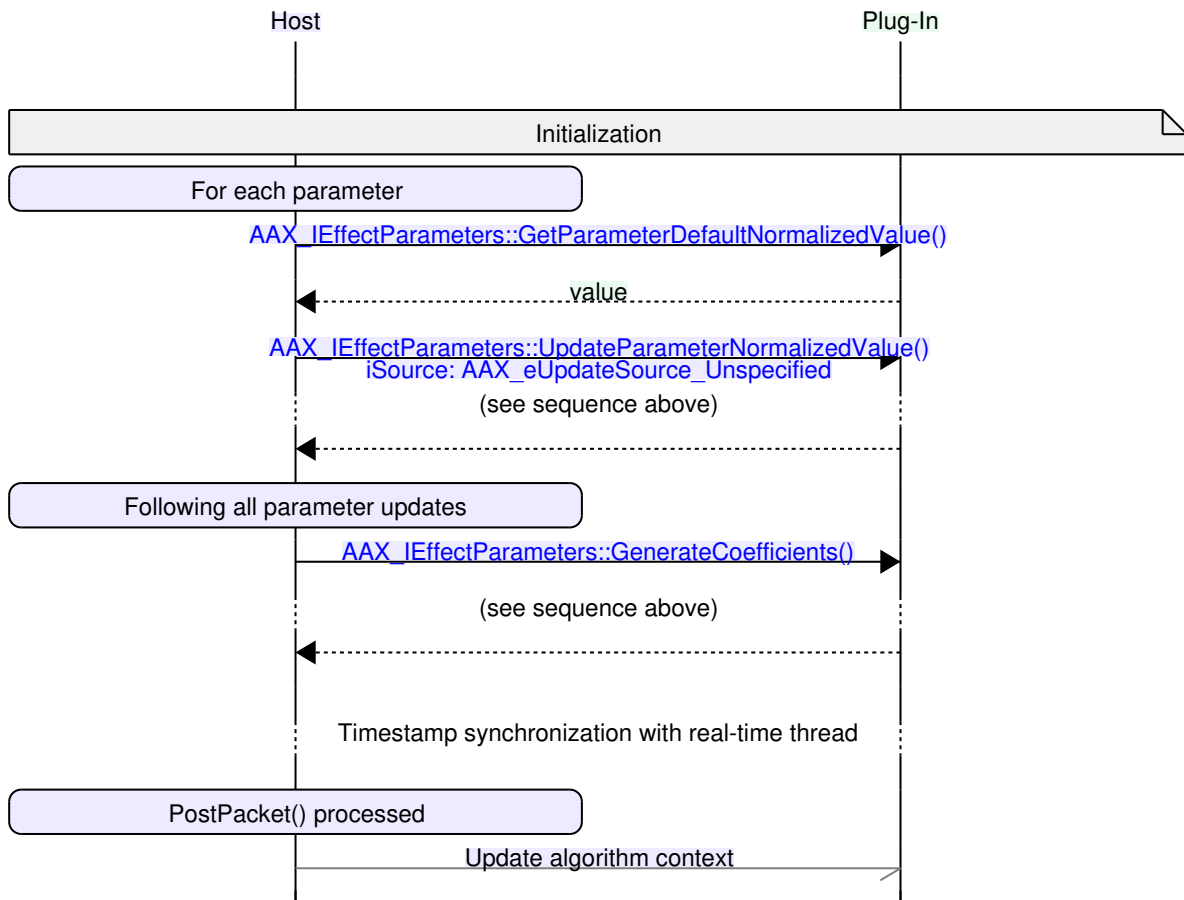
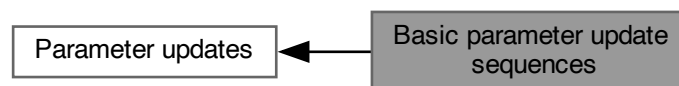


Figure 12.11 Sequence of method calls and events for parameter updates at plug-in initialization

Collaboration diagram for Basic parameter update sequences:



12.34 Linked parameters

How to link parameters.

12.34.1 On this page

- [Basics of Linked Parameters](#)
- [Linked Parameter Operation](#)
- [Changing Tapers](#)

12.34.2 Basics of Linked Parameters

A "linked" parameter can be defined as any parameter whose state is somehow dependent on another parameter. Within this general definition, there are various different kinds of parameter linking:

- Linking behavior can operate one-way or parameters can be reciprocally linked
- Linking between parameters can be one-to-one or one-to-many

12.34.2.1 Basic considerations for parameter linking

Although the concept of parameter linking is simple, implementing linked parameter behavior that is intuitive and consistent can require careful design.

- Parameter interdependencies and constraints can become complex, especially when handling multiple sets of linked parameters.
- Linked parameters that update other parameters during playback can result in subtle timing inconsistencies
- Automated parameters may contain arbitrary and conflicting automation data
- A user may attempt to edit multiple linked parameters simultaneously during playback, e.g. using multiple encoders on a control surface
- A plug-in may contain dependency cycles between interdependent parameters. These cycles can cause undesired behavior that is difficult to debug, especially if it only occurs in certain circumstances such as when loading particular presets.

In all of these cases, a plug-in should provide consistent linked parameter behavior: every automated playback pass should be identical, parameters should never "fight" one another or trigger rapid and unexpected changes in other parameters, parameters should not become "stuck" in a particular state, etc.

Here comes trouble

12.34.2.2 Defining proper linked parameter behavior

A good way to approach parameter linking is to start with an understanding of exactly what behavior you desire.

Here are some behaviors that you probably *don't* want in your plug-in:

- BAD: Parameters are only linked when edited from the plug-in GUI. Users may attempt to edit linked parameters from attached control surfaces or using the host's automation features. The parameters should behave the same way regardless of which method is used to edit them.
- BAD: Parameters try to match all automation data. Automation data can be written arbitrarily: Pro Tools doesn't have any restrictions that a user with a pencil tool must draw inside the lines, or a user may attempt to edit multiple parameters on an attached control surface simultaneously. Any parameter that attempts to match both its own automation data and the automation data of another parameter, or any parameter that attempts to set another automatable parameter's state based on its own automation data, will lead to "fighting" during playback of non-conformant automation.
- BAD: Automation data is only written to one lane at a time. One approach to parameter linking may be to only write automation data to a single parameter at a time. This could be the parameter that is currently being touched and edited, or it could be a dedicated "master" parameter within the linked group. While this approach can be used to solve some types of conflicts, it can still lead to unnecessarily complex or inconsistent behavior in certain situations: for example, arbitrary automation data can still be written to multiple parameters' automation lanes, or a user can choose to record automation for only one parameter in a set but can skip the "master" parameter. Furthermore, it is difficult if not impossible to properly handle parameters that can be dynamically linked or un-linked using this approach.

With those potential problems in mind, here is a description of how parameter linking *should* behave.

12.34.2.2.1 Correct behavior for linked parameters *Notes*

- In this proposal (and throughout the rest of this page) the term "linker" will refer to a parameter that initiates a change and the term "linked" will refer to a dependent parameter that receives the change.
- The following discussion will focus on *automatable* parameters that are *reciprocally linked*. This case tends to be the most complex, with the greatest need for consistency of implementation.

During user-generated real-time edits (from the plug-in GUI or a control surface) both the linker and the linked parameters should be updated. Without this requirement, there would be no parameter linking. In order for this requirement to be enforced consistently the following behaviors must be maintained:

- The linked parameter should not jump to a new value if the user attempts to edit both parameters simultaneously using a control surface.
- To ensure proper automation playback, automation should be written to both the linker and the linked parameters.

When playing back automation, parameters should operate independently and should not attempt to force dependent parameters to a new state. This prevents fighting in the presence of incompatible automation and ensures deterministic automation playback with every playback pass. As above, there are some more subtle behaviors that must be maintained for this to work properly:

- If the user begins a real-time edit during automation playback then the parameter linking behavior should resume as described above

- If the plug-in's algorithm cannot support certain parameter configurations then its automatable parameters should be decoupled from the algorithm using a set of coefficients that is aware of the algorithm's constraints. In this way every combination of parameter states can map to a particular coefficient state, maintaining determinism, and incompatible parameter combinations can simply resolve to the "closest" match in the possible coefficient space during playback of edited parameter automation data.
- Another, simpler approach for plug-ins that do not support arbitrary parameter configurations is to ensure that the problematic parameters are not automatable. Handling non-automatable parameter linking is much easier in general, so consider this approach if automation is not a requirement for some of your plug-in's parameters.

When handling preset changes and plug-in initialization, a similar approach should be taken as with plug-in automation playback. In these cases it is very unlikely that the plug-in's parameters will be left in an incompatible state and attempts at linking may result in unwanted update cycles between inter-dependent parameters or unnecessary coefficient churn. This latter concern can be a real problem for AAX DSP plug-ins that initialize internal algorithmic state based on initial coefficient data.

12.34.2.2.2 Compatibility caveat This behavior was not possible under the RTAS/TDM format, and many RTAS and TDM plug-ins reverted to workarounds such as writing automation to only one parameter at a time and linking the parameters during playback. Therefore, plug-ins that previously supported linked automatable parameters under the RTAS/TDM format may not be able to both implement this recommended parameter linking behavior and maintain compatibility with automation in saved sessions.

Most of Avid's plug-ins that were available in the RTAS and TDM formats fall into this category and should not be used as examples of proper parameter linking behavior. Instead, use the SDK's DemoGain_LinkedParameters example plug-in as an example of proper linked parameter operation.

12.34.3 Linked Parameter Operation

As described above, the key rule for linked parameters is to link during real-time user edits *only*, and should operate the parameters independently (without linked behavior) during automation playback and preset restore. This rule will simplify many issues: it will prevent conflicts with automation data, avoid potentially strange behaviors when restoring presets, and more.

Here is how the system works WITH linked parameters, using code snippets from the DemoGain_LinkedParameters example plug-in:

12.34.3.1 User Editing

1. User clicks on a parameter in the GUI or grabs a parameter on the controls surface. A TOUCH token should be sent at this point.
 - The touched parameter status comes back to the plug-in. If the parameters are linked the other linked parameter should have a TOUCH token sent. This really should only be done for linked continuous parameters. This is done by overriding the `AAX_CEffectParameters::UpdateParameterTouch()` method.
2. The user changes the parameter from the GUI or controls surface. A SET token should be sent at this point.

```
// *****
// METHOD: UpdateParameterTouch
// *****
AAX_Result DemoGain_Parameters::UpdateParameterTouch ( AAX_CParamID inParameterID, AAX_CBoolean
inTouchState )
{
    if ( inTouchState )
    {
        AAX_CParamID linkedControl = this->GetLinkedControl ( inParameterID );
        if ( linkedControl )
        {
            this->TouchParameter ( linkedControl );
            mLinkTouchMap.insert ( std::pair<std::string, std::string>( inParameterID, linkedControl ) );
        }
    }
    [...]
}
```


3. The SET token goes into the system and comes back to the plugin via [AAX_CEffectParameters::UpdateParameterNormalizedValue\(\)](#)

- If the parameter is linked then the other linked parameter should have its value set for its linked behaviour. The system knows this is a linked parameter so when the value comes back to the plug-in via `UpdateParameterNormalizedValue()` it will know not to perform linked behaviors on that value change. To determine if a parameter should set a linked parameter you check it with the [AAX_CEffectParameters::IsParameterTouched\(\)](#) method.

4. The plug-in updates its internal state and sends an UPDATE tokens for both parameters.

```
// *****
// METHOD: UpdateParameterNormalizedValue
// *****
AAX_Result DemoGain_Parameters::UpdateParameterNormalizedValue ( AAX_CParamID inParameterID, double
inValue, AAX_EUpdateSource inSource )
{
    AAX_Result result = AAX_CEffectParameters::UpdateParameterNormalizedValue ( inParameterID,
inValue, inSource );
    bool touched = this->IsParameterTouched ( inParameterID );

    [...]
        if ( touched && inSource == AAX_eUpdateSource_Unspecified )
        {
            if ( type == eType_Pan )
                this->SetParameterNormalizedValue( linkedControl, (1.0 - inValue) );
            else if ( type == eType_Gain )
                this->SetParameterNormalizedValue( linkedControl, inValue );
        }
    [...]
}
```

5. Repeat steps 2-4 while changing the parameter.

6. The user lets go of the GUI or controls surface. A TOUCH token with the released state should be sent.

- The touched parameter status comes back to the plug-in. If the parameters were linked the other linked parameter should have a TOUCH token with the release status sent. This again is done by overriding the [AAX_CEffectParameters::UpdateParameterTouch\(\)](#) method.

```
// *****
// METHOD: UpdateParameterTouch
// *****
AAX_Result DemoGain_Parameters::UpdateParameterTouch ( AAX_CParamID inParameterID, AAX_CBoolean
inTouchState )
{
    if ( inTouchState )
    {
        [...]
    }
    else
    {
        [...]
        this->ReleaseParameter ( iter->second.c_str () );
        [...]
    }

    return AAX_SUCCESS;
}
```

12.34.3.2 Automation Playback

1. The SET token comes from the automation system and enters the plugin via [UpdateParameterNormalizedValue\(\)](#).

- The plug-in will know this is not from the user editing therefore it will NOT set the other linked parameter. Remember ONLY LINK USER EDITING. That way there's no conflicts if the user edited the automation or if the order in which automation arrives at the plug-in changes.

2. The plug-in updates its internal state and sends an UPDATE token.

3. Repeat steps 1-2 while playing back automation.

12.34.3.3 Chunk Restoring

1. Plug-in loads the chunk.
2. The plug-in sets every parameters value.
3. The SET tokens comes back to the plugin via UpdateParameterNormalizedValue().
 - The plug-in will know this is not from the user editing therefore it will NOT set the other linked parameter. Remember ONLY LINK USER EDITING. Hopefully the result of this is that the contents of the chunk will be restored to its exact state.
4. The plug-in updates its internal state and sends out UPDATE tokens.

12.34.4 Changing Tapers

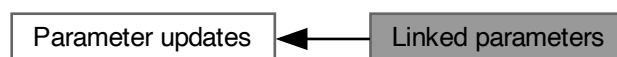
One common use of linked parameters is to change the taper associated with a parameter. For changing tapers there are basically only a two rules you need to follow:

1. When you're loading a new chunk you need to set the taper values first. If a parameter is what updates the taper then set that value first. That way when the value of a parameter is set from a chunk it wont change because of a taper change.
2. Update the taper from the UpdateParameterNormalizedValue() method. If the new taper needs to change the value of the parameter you only do so if the user is editing the linked parameter. This still follows the ONLY LINK USER EDITING rule.

```
AAX_Result Simple_Parameters::UpdateParameterNormalizedValue ( AAX_CParamID inParameterID, double inValue,
    AAX_EUpdateSource inSource )
{
    // GetLinkedControl() is a user defined method which determines the linked control ID.
    AAX_CParamID linkedControl = this->GetLinkedControl ( inParameterID );
    if ( linkedControl )
    {
        // IsParameterLinkReady()* is a built in method of AAX_CEffectParameters which determines if the
        // parameter should perform linked behaviors based on the touch state of the parameter and the
        // source of the UpdateParameterNormalizedValue() call.
        if ( this->IsParameterLinkReady ( inParameterID, inSource ) )
            this->SetParameterNormalizedValue( linkedControl, inValue );
    }

    // Call the inherited method for the original parameter
    AAX_Result result = AAX_CEffectParameters::UpdateParameterNormalizedValue ( inParameterID, inValue,
        inSource );
    return result;
}
```

Collaboration diagram for Linked parameters:



12.35 Linked parameter update sequences

Sequence diagrams for some common linked parameter update scenarios.

12.35.1 On this page

- [User-generated update](#)
- [Update from automation playback](#)

Note

To enable logging for these events at run time set `DTF_AUTOMATION=file@DTP_LOW` in the [DigiTrace](#) configuration file.

12.35.1.1 Notes on threading for these sequences

- Calls from the host into [AAX_IEffectParameters](#) may occur on any thread. In general, the only synchronization that is guaranteed for data model calls in these diagrams is that the call will follow whatever event is indicated as its trigger.
- Calls from the host into [AAX_IEffectGUI](#) will occur on the main application thread unless indicated otherwise in the [AAX_IEffectGUI](#) documentation.
- Host-driven updates to the algorithm context are always synchronized with the real-time processing thread

See also

[Basic parameter update sequences](#)

12.35.2 User-generated update

This is the sequence of calls for a parameter update triggered by the user. For this sequence, we assume that the edit was triggered by a GUI event. Updates from control surfaces are handled in exactly the same way, except that the parameter touch, set value, and release tokens are generated by the control surface.

In this example the updated parameter is reciprocally linked to one other parameter. These are the "linker" and "linked" parameters, respectively.

This procedure is very similar to the non-linked case described [here](#). In the diagrams below, red arcs and pink section headings are used to indicate events that are specific to the linked parameter case.

Notes:

1. This sequence shows the linked parameter reciprocally issuing a touch on the linker parameter. The touch fails since the linker parameter is already touched at this time. If the roles were reversed (if an edit occurred on the linked parameter) then this touch would succeed.
2. The host flags all set value tokens that are triggered by a plug-in within the scope of [AAX_IEffectParameters::UpdateParameterNormalizedValue\(\)](#). When those set value tokens are processed they result in additional calls to [AAX_IEffectParameters::UpdateParameterNormalizedValue\(\)](#) "UpdateParameterNormalizedValue()". The host sets `iSource` to [AAX_eUpdateSource_Parameter](#) for each of these subsequent calls to indicate that the update originated from within a parameter update event.
3. [IsParameterLinkReady\(\)](#) returns `true` during the linker parameter update because the update source is unknown and the parameter is touched. Both conditions must be true in order for the linking logic to proceed with setting linked parameters' values.
4. [IsParameterLinkReady\(\)](#) returns `false` during the linked parameter update because the source is [AAX_eUpdateSource_Parameter](#). This prevents update cycles for reciprocally linked parameters, as demonstrated here.

12.35.2.1 High-level interface calls and events

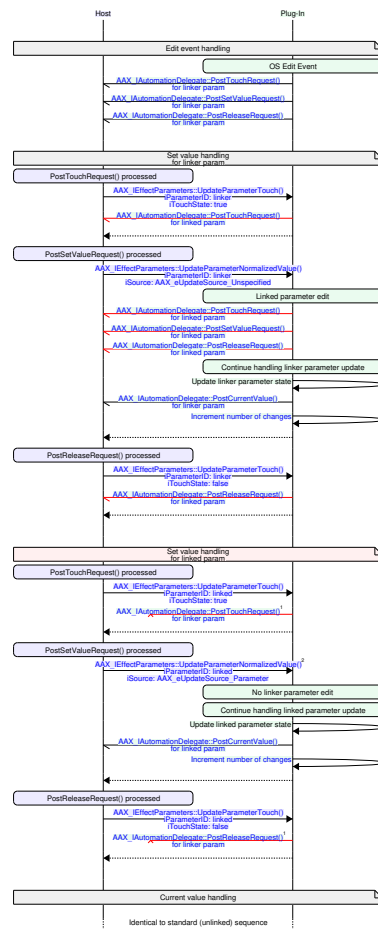


Figure 12.12 High-level sequence of interface calls and events for a reciprocally linked parameter update following a user-generated edit

12.35.2.2 Detailed interface calls and events

Note that this diagram assumes a GUI implementation that uses [SetParameterNormalizedValue\(\)](#). The implementation could also use other parameter set methods, either in [AAX_IEffectParameters](#) or directly on an [AAX_IParameter](#). The overall sequence would remain the same.

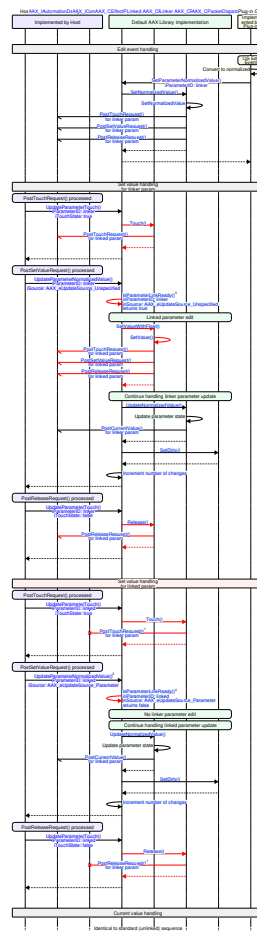


Figure 12.13 Detailed sequence of method calls and events for a reciprocally linked parameter update following a user-generated edit on the plug-in GUI

12.35.3 Update from automation playback

Since all parameter linking occurs while recording automation, automation playback is very simple. The automation lanes may contain any arbitrary values, so, in order to avoid fighting between incompatible values, the plug-in should respect all automation values during playback.

Notes:

1. `IsParameterLinkReady()` returns `false` during automation playback because the updated parameter is not touched. This ensures that automation playback will proceed with the written values and also guarantees that the user will always be able to override the automation using a control surface encoder or GUI editor.

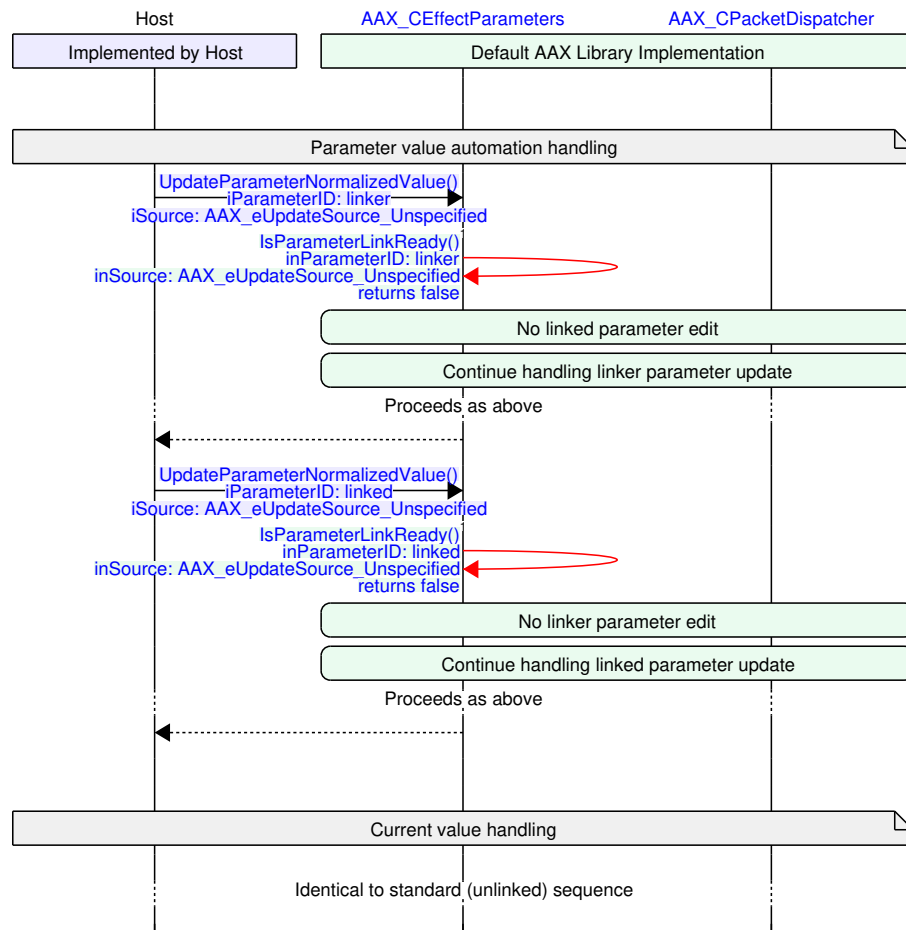
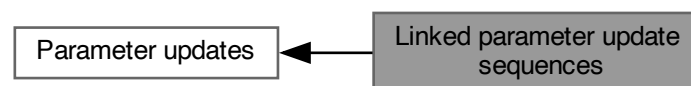


Figure 12.14 Sequence of method calls and events during automation playback with linked parameters

Collaboration diagram for Linked parameter update sequences:



12.36 Plug-in type conversion

Specification for valid conversions between plug-in types.

12.36.1 About this specification

The specification on this page defines the valid AAX plug-in type conversions. An AAX host may use this specification to perform automatic type conversions. For example:

- When a session that was saved with a DSP plug-in Type is opened on a Native system the saved DSP Type should be converted to its Native counterpart.
- When a session saved with the free version of a plug-in is opened on a system that has the full version installed then the saved free Type may be converted to the full version.

12.36.2 Terminology

In this specification the term "Type" refers to a specific configuration of an AAX plug-in.

Each Type is uniquely identified by a combination of five values:

- Manufacturer ID
- Product ID
- Plug-In ID
- Sample rate bit mask
- Architecture

Each Type is defined by a particular call to [AAX_IComponentDescriptor::AddProcessProc_Native](#) or [AAX_IComponentDescriptor::AddProcessProc_TI](#) in the plug-in's description method.

The Plug-In ID is defined as one of [AAX_eProperty_PluginID_Native](#) or [AAX_eProperty_PluginID_TI](#).

In this specification, the format for describing an ID is:

[ID triad + sample rate + architecture]

Where the ID triad may be expanded to [[Manufacturer ID] [Product ID] [Plug-In ID]]

- Explicit values are given in plain face
- Arbitrary constant values are given in *bold face*
- Wildcard values are given in *italics*

For example:

- [*ID triad* + *any sample rate* + Native]
 - Defines all Type identifiers with the same ID triad and the Native architecture, regardless of sample rate.
- [[*Manufacturer ID*] [*Product ID*] [*any ID*] + *sample rate* + Native]
 - Defines all Type identifiers with the same Manufacturer and Product IDs, the same sample rate, and the Native architecture, but with any plug-in ID.

12.36.3 Scope of this specification

For the purposes of this specification we are not concerned with AudioSuite Types. Currently these Types are never type-converted.

In theory, an AAX host may apply a "partial" type conversion by swapping between different ProcessProcs without destroying and re-building any of the plug-in's objects (data model, GUI). For the purposes of this specification we are not concerned about whether a given conversion is "partial" or "total"; all conversions are treated the same.

Plug-in conversions are also required between Types of different stem formats. In fact, the supported stem format is an integral part of a Type's unique identifier. This specification ignores the question of stem formats entirely; we assume that each Type supports all necessary stem formats for legal conversion with other Types.

In general, type swapping rules for deprecated types and related types are equivalent. See [Type deprecation](#) for more information about the differences between related and deprecated types.

12.36.4 Topological constraints

- All Types included in a single [AAX_ICollection](#) must have the same Manufacturer ID
- All Types included in a single [AAX_IEffectDescriptor](#) must have the same Manufacturer ID and Product ID
- The Sample rate bit masks for two Types that share all other identifiers must be non-overlapping. I.e. `0x0 == sr_mask1 & sr_mask2`
- Two Types may not be registered that differ only in their Architecture
- Type relationships may only be defined between mutually exclusive Types. Types are mutually exclusive if:
 - Their sample rate support is non-overlapping
 - The "before" Type's ID triad is not associated with any Type in the plug-in

12.36.5 Implicit conversions

The AAX host should automatically convert between Types within the following IDs:

- [*ID triad* + *any sample rate* + *Architecture*]
- [[[*Manufacturer ID*] [*Product ID*] [*any ID*]] + *sample rate* + *any architecture*]

These conversions occur only if both Types are included in the plug-in when the conversion is made. Consider the following scenario:

1. A session is saved including an plug-in instance with the following Type identifier: [My ID triad + 48 kHz + TI]
2. The plug-in is updated to a version that does not include this Type identifier, but that does include [My ID triad + 48 kHz + Native]
3. The session is opened on a Native system

In this scenario, if the plug-in had not been updated then an automatic conversion would occur. However, since the plug-in no longer includes the saved Type identifier, no automatic conversion occurs.

A plug-in can work around this situation by including the "old" Type identifier as a Related (or Deprecated) Type (see [Explicit conversions](#).)

When a plug-in instance is saved after making an implicit conversion, plug-ins may be saved with the session using their new Type identifier. This is not required. For example, Pro Tools will prefer to save plug-in instances that were converted from DSP types as DSP, even if they were converted to corresponding Native types when the session was loaded onto and saved from a native Pro Tools system.

12.36.6 Explicit conversions

AAX includes properties that allow a plug-in to define explicit relationships between different Types. These properties only operate on ID triads. Each property can be associated with an array of ID triads to define a one-to-one or a many-to-one association between the given ID triads and the specific Type to which the property is attached.

- [AAX_eProperty_Related_DSP_Plugin_List](#) / [AAX_eProperty_Deprecated_DSP_Plugin_List](#)
 - All of the given ID triads specify TI Types
- [AAX_eProperty_Related_Native_Plugin_List](#) / [AAX_eProperty_Deprecated_Native_Plugin_List](#)
 - All of the given ID triads specify Native Types

The AAX host should convert between related Types within the following constraints:

- [[[Manufacturer ID] [Related Product ID] [Related TI Plug-In ID]] + *sample rate* + TI]
 - -> [[[Manufacturer ID] [New Product ID | Related Product ID] [New Plug-In ID]] + *sample rate* + *any architecture*]
- [[[Manufacturer ID] [Related Product ID] [Related Native Plug-In ID]] + *sample rate* + Native]
 - -> [[[Manufacturer ID] [New Product ID | Related Product ID] [New Plug-In ID]] + *sample rate* + *any architecture*]

These conversions will occur regardless of whether the related ID triad is used for any of the Types in the plug-in.

When a plug-in instance is saved after making an explicit conversion, all plug-ins are saved with the session using their new Type identifier.

Host Compatibility Notes Pro Tools versions prior to Pro Tools 12.3 do not allow explicit type conversion between types with different product ID values. Beginning in Pro Tools 12.3 both the product ID and the plug-in ID may differ between explicitly related types.

12.36.7 Type deprecation

There are two varieties of explicit plug-in type association: related types and deprecated types.

Properties that create a related type association:

- [AAX_eProperty_Related_Native_Plugin_List](#)
- [AAX_eProperty_Related_DSP_Plugin_List](#)

Properties that create a deprecated type association:

- [AAX_eProperty_Deprecated_Native_Plugin_List](#)
- [AAX_eProperty_Deprecated_DSP_Plugin_List](#)
- [AAX_eProperty_PluginID_Deprecated](#)

With a few exceptions, these two types of explicit association are treated identically. These are the ways in which deprecated plug-in type association differs from related plug-in type association:

- If plug-in types A and B are both installed, and if type A deprecates type B, then type B will be excluded from all insert lists in Pro Tools and will be made invisible to the user.
- If plug-in types A and B are both installed, and if type A deprecates type B, then instances of type B will be swapped to type A when a session containing saved instances of type B is opened.
- Upon the next session save following a swap due to a deprecated type association, the plug-in instance will be saved into the session using the ID of the new plug-in type. Therefore deprecated type swaps do not round-trip when moving between multiple systems if some of the systems have the deprecated types, but not the deprecating types, installed.

Type deprecation should only be used when a new version of an effect completely replaces an old version of the effect. For all other situations, type relationship should be used.

Host Compatibility Notes

- Pro Tools versions before Pro Tools 12.3 treat deprecated and related type associations identically and do not support type deprecation features
- Media Composer does not support type deprecation features
- VENUE does not support type deprecation features

Collaboration diagram for Plug-in type conversion:



12.37 The Avid Component Framework (ACF)

12.37.1

How the AAX C++ interfaces work.

The objects and interfaces in AAX are based on the Avid Component Framework (ACF). The ACF is Avid's implementation of COM, and is the framework that AAX, as well as AVX (Avid Video Extensions) plug-ins are built on.

ACF can be considered an implementation detail of the AAX SDK; the SDK is written to protect plug-in developers from the intricacies of ACF, and it is not necessary to understand ACF or COM in order to use the SDK.

12.37.2 More details

As in COM, ACF draws a distinction between the concept of an object and the concept of an interface. An object is treated as a "black box" of code, whereas an interface is a class of pure virtual methods that allows one to access the functionality inside the object. An object in ACF is represented by the [IACFUnknown](#) interface, which is binary compatible with the COM class IUnknown. (Likewise, [IACFUnknown](#) follows the same reference counting rules as IUnknown objects.) This interface allows a client to get pointers to other interfaces on a given object using the [QueryInterface\(\)](#) method.

Reference counting is an important aspect of both COM and ACF. Simply put, reference counting is the practice of tracking all references to an object, so that a program can determine when the object can safely be deleted. The AAX SDK library handles this reference counting behind the scenes, so plug-ins that call into the SDK library to manage their component interfaces will not leak references.

Many additional resources can be found both online and print that cover COM and reference counting in greater detail.

12.37.3 ACF interfaces in AAX

The binary interface between an AAX plug-in and host is defined by a series of ACF interfaces. Each of these interfaces inherits from [IACFUnknown](#). The implementation of each ACF interface typically uses [CACFUnknown](#), a utility class that provides basic reference counting and additional fundamental ACF details to satisfy [IACFUnknown](#).

These ACF interfaces may be implemented by either the AAX plug-in or the host. The host retains a reference to each interface that is implemented by the plug-in in order to call methods on the plug-in's implementation. Correspondingly, the plug-in retains references to various interfaces that are implemented by the host, and may call host methods via these interfaces.

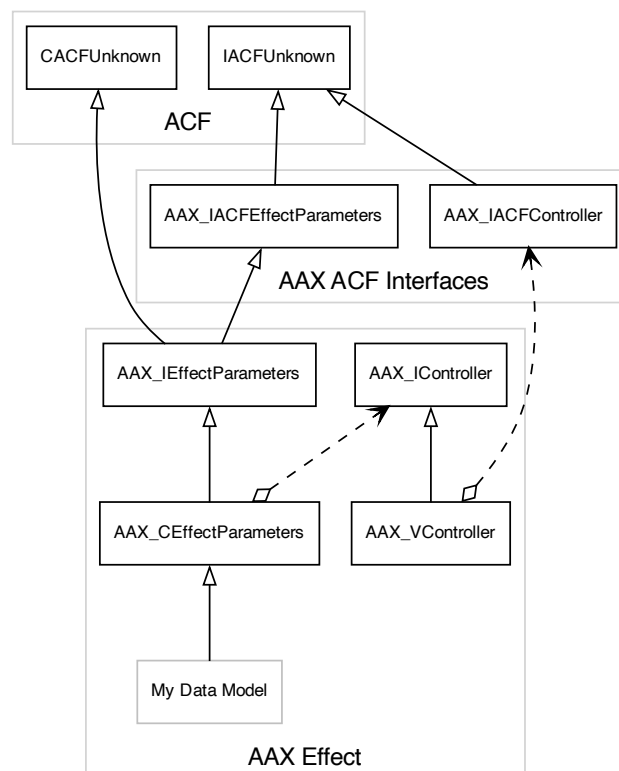


Figure 12.15 ACF interfaces: **AAX_IACFEffParameters** and **AAX_IACFController**

The figure above demonstrates this design: the plug-in implements `AAX_IACFEffParameters` directly, and retains a reference to an `AAX_IACFController` that is implemented by the host.

In order to implement `AAX_IACFEffParameters`, `AAX_IEffectParameters` inherits from `CACFUnknown` and implements `QueryInterface()` to ensure that the `IACFUnknown` interface is implemented. The rest of the implementation of `AAX_IACFEffParameters` is contained in `AAX_CEffectParameters` and the plug-in's custom data model class.

The reference to `AAX_IACFController` is managed by a versioned implementation class. For more information about this design, see below.

12.37.4 Using ACF interfaces

Depending on where an interface is implemented, there are two specific ways to acquire a reference to the underlying AAX object from an `IACFUnknown` pointer:

- [Host-provided interfaces](#)
- [Plug-in interfaces](#)

12.37.4.1 Host-provided interfaces

Interfaces that are managed by the host must be carefully version-controlled in order to maintain compatibility with many different host versions. The AAX SDK includes "AAX_V" classes to handle this versioning. `AAX_V` classes are concrete classes that query the host for the correct version of the requested interface. These classes can also handle re-routing deprecated calls and other complicated versioning logic.

To create an `AAX_V` object, pass an `IACFUnknown` pointer to the underlying host-managed interface in to the `AAX_V` class' constructor. ACF reference counting is handled automatically by the object's construction and destruction routines, so no additional calls are necessary to acquire and release the reference.

```
#include "AAX_VController.h"

void SomeFunction (IACFUnknown * inController)
{
    // When object is created, a reference is acquired
    AAX_VController theController (inController);

    //
    // ...
    //

    // When object goes out of scope, the reference is released
}
```

12.37.4.2 Plug-in interfaces

Interfaces to objects that are owned by the plug-in always have a known version and therefore do not require `AAX_V` object management. Instead, these interfaces must be acquired and released directly using ACF.

```
#include "AAX_UIDs.h"
#include "AAX_Assert.h"
#include "AAX_IEffectParameters.h"
#include "acfunknown.h"

void SomeFunction (IACFUnknown * inController)
{
    // When interface is queried, a reference is acquired
    if ( inController )
    {
        AAX_IEffectParameters* myEffectParameters = NULL;
        ACFRESULT acfErr = ACF_OK;
        acfErr = inController->QueryInterface(
            IID_IAAXEffectParametersV1,
```

```

        (void **) &myEffectParameters);

    AAX_ASSERT(ACFSUCCEEDED(acfErr));
}

//
// ...
//

// The reference must be explicitly released when finished
if (myEffectParameters)
{
    myEffectParameters->Release();
    myEffectParameters = NULL;
}
}

```

12.37.5 Interface versioning in AAX

The ACF-based interface used by AAX is designed to allow additional features to be added to the architecture. This can be achieved via the addition of new kinds of interfaces (e.g. [AAX_IEffectDirectData](#)) or by extending the existing interfaces. In this section, we will describe an approach for interface extension.

First, here is a more complete picture of "version 1" of the [AAX_IACFEffParameters](#) and [AAX_IACFController](#) interfaces, including a possible host implementation of [AAX_IACFController](#) :

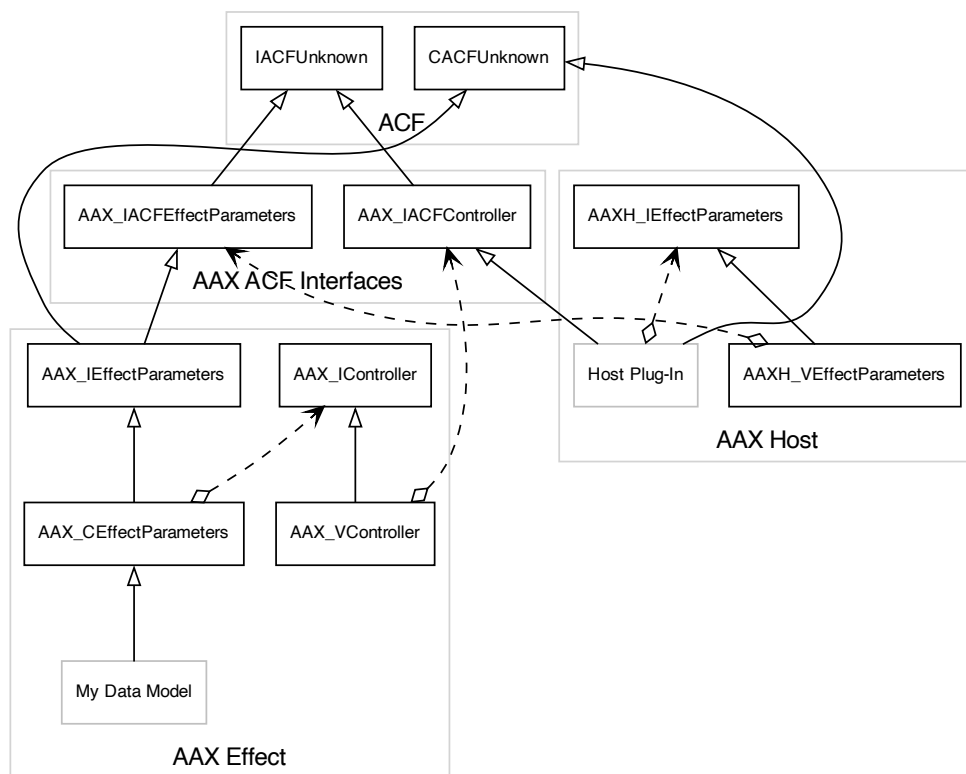


Figure 12.16 ACF interfaces: [AAX_IACFEffParameters](#) and [AAX_IACFController](#) (with possible host design)

To extend these interfaces, new "version 2" interfaces are created that inherit from the original interface classes. Although any version 1 method could be called on the new version 2 class, references to each interface are retained by the client in order to clarify the specific version in which each method was introduced.

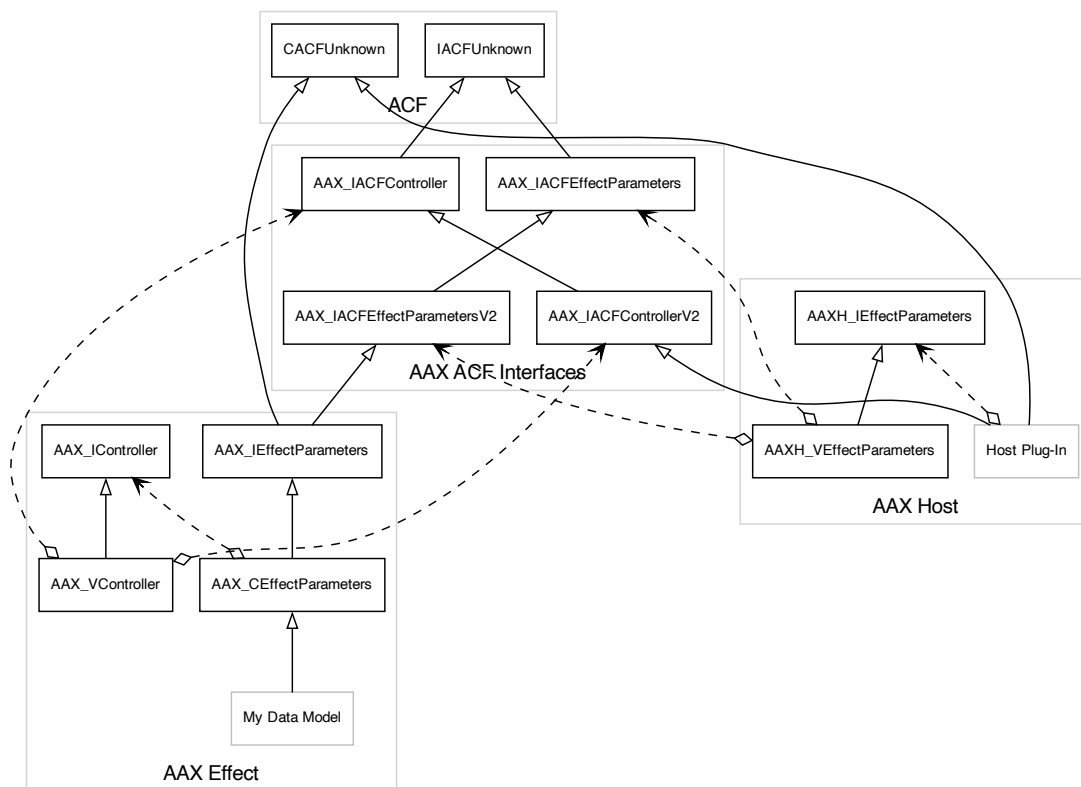


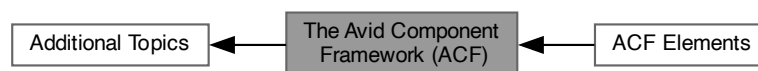
Figure 12.18 Complete design example with versioned ACF interfaces

Documents

- [ACF Elements](#)

ACF classes that are used by common AAX interfaces.

Collaboration diagram for The Avid Component Framework (ACF):



12.38 ACF Elements

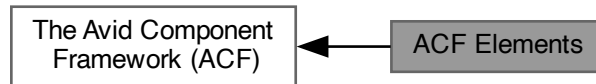
12.38.1

ACF classes that are used by common AAX interfaces.

Classes

- interface [IACFUnknown](#)
COM compatible IUnknown C++ interface.

Collaboration diagram for ACF Elements:



12.39 AAX Host Guides

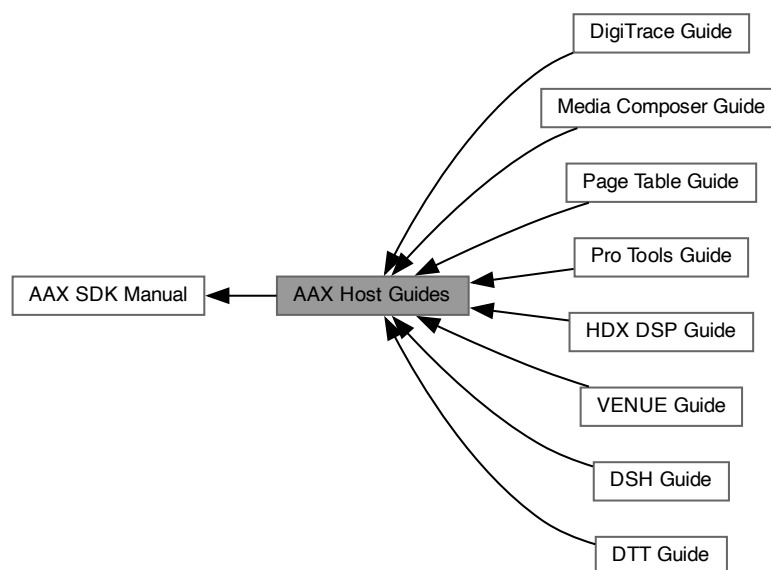
12.39.1

Documentation for specific AAX host environments.

Documents

- [Pro Tools Guide](#)
Details about using AAX plug-ins in Pro Tools.
- [Media Composer Guide](#)
Details about using AAX plug-ins in Media Composer.
- [HDX DSP Guide](#)
How to write AAX plug-ins for Avid's TI DSP-based platforms.
- [Page Table Guide](#)
How to map a plug-in's parameters to control surfaces.
- [DigiTrace Guide](#)
How to add tracing to your plug-ins and view logging from the plug-in host.
- [DSH Guide](#)
How to test basic functionality of AAX plug-ins using DSH test tool.
- [DTT Guide](#)
How to automate different test scenarios for DSH.
- [VENUE Guide](#)
Details about using AAX plug-ins in VENUE live sound systems.

Collaboration diagram for AAX Host Guides:



12.40 Pro Tools Guide

Details about using AAX plug-ins in Pro Tools.

12.40.1 Contents

- [About this document](#)
- [Processing modes](#)
- [Requirements for AAX plug-in compatibility with Pro Tools](#)
- [Audio Engine Behavior and Features](#)
- [Basic plug-in operation](#)
- [Optional plug-in features](#)
- [Using the Pro Tools Scripting SDK with AAX](#)
- [Debugging AAX plug-ins](#)
- [Troubleshooting common AAX plug-in failures](#)
- [Using DigiOptions](#)
- [Compatibility Notes](#)

12.40.2 About this document

This guide discusses specific details related to using AAX plug-ins with Pro Tools, such as loading and initialization procedures, GUI hosting, and other application-specific features. This guide is not intended to provide complete documentation for the Pro Tools application. For more information about the features, functionality, and use of Pro Tools see the Pro Tools Reference Guide, available for download from the Avid web site.

This guide is a work in progress, and is extended as needed to describe different aspects of and caveats to the Pro Tools implementation of the AAX host spec.

12.40.3 Processing modes

Pro Tools supports three AAX processing modes: AudioSuite, AAX Native, and AAX DSP.

- AudioSuite plug-ins perform non-real-time, random-access, file-based processing entirely on the host CPU.
- AAX Native plug-ins perform real-time, linear, non-destructive processing entirely on the host CPU. Native plug-ins are also used by Pro Tools to perform offline rendering.
- DSP plug-ins perform real-time, linear, non-destructive processing on DSP-accelerated hardware, with non-real-time tasks performed on the host CPU.

Each of these processing modes offers specific advantages and trade-offs in functionality, power, and development effort, and plug-in developers may choose to develop only for specific processing modes if the features provided by those modes are required by the plug-in.

12.40.3.1 Real-time processing

Real-time processing allows users to operate plug-ins in live signal paths or in complicated audio routing schemes when the future input data is not known.

Plug-ins operating in real-time are clients of the Pro Tools automation system, meaning that control movements can be dynamically recorded and played back with the audio track, written by hand onto the Pro Tools timeline for future playback, and/or edited by and broadcast to attached control surfaces.

To instantiate a plug-in for real-time processing in Pro Tools, click on an insert slot in the desired track and select the plug-in from the menu that appears

12.40.3.1.1 Native real-time processing When an AAX plug-in is run natively, all of its components and processing elements are loaded into the host environment. The host CPU handles the plug-in's real-time audio processing as well as its data model, GUI, and other tasks.

12.40.3.1.2 DSP real-time processing When an AAX plug-in is run with Avid's DSP-enabled hardware, the plug-in's real-time processing code is loaded onto the external DSP device, while the remaining plug-in modules continue to be run by the host CPU. Each DSP in this system provides dedicated processing capacity that is not shared with an OS or other processes, and therefore this architecture allows users to achieve highly reliable and deterministic low-latency processing even when many DSP plug-ins are instantiated.

Figure 1: Real-time insert slots in the Pro Tools Edit and Mix windows.

12.40.3.1.3 CPU reporting To guarantee absolute reliability, AAX DSP plug-ins are required to report their worst-case performance metrics to Pro Tools. Pro Tools uses this information to ensure that each DSP in the system will be loaded with only the number of plug-ins that it can support given a worst-case processing load.

12.40.3.2 Non-real-time processing (AudioSuite)

The non-real-time AudioSuite processing mode is file-based, meaning that the results of AudioSuite processing are applied destructively to audio files (generally to new, empty files provided by Pro Tools.) AudioSuite processing can only be performed on preexisting blocks of audio.

There are two primary ways to apply AudioSuite processing in Pro Tools. The first way is to selectively apply the processing algorithm to the audio tracks and clips that are selected on the Pro Tools timeline. This is known as "destructive" processing, because the original audio track is replaced by the new processed audio track. There are no limitations governing the amount of time required to process a track in this manner.

Audio Suite plug-ins also have a second optional mode in which they can run. This is referred to as Preview mode. The Preview feature allows you to monitor the audio processing applied to a track in semi-real-time. Because this is a real-time process, it is not applicable to all types of file based processing. You may elect not to support this mode in your plug-in if its algorithm does not lend itself to real-time, linear processing. Preview mode is implemented in a non-destructive manner, as Preview mode exists for auditioning only with no actual replacement of audio data on the Pro Tools timeline.

To instantiate an AudioSuite plug-in in Pro Tools, select the plug-in from the "AudioSuite" menu in the Pro Tools application menu bar

12.40.3.3 Multichannel and Multi-Mono

Pro Tools supports various surround stem formats throughout the entire signal chain, including multi-channel processing through AAX plug-ins.

Pro Tools also allows a plug-in to function in multi-mono mode if the plug-in does not explicitly support certain channel formats. In multi-mono mode, Pro Tools instantiates a separate instance of a plug-in for every channel in the track. In this mode, plug-in controls across all channel-instances in a multi-mono collection are linked by default, though channels can be unlinked by toggling the blue link button in the plug-in header and selecting the channel whose controls you wish to modify.

For more information about multi-mono mode, please refer to the Pro Tools Reference Guide.

12.40.4 Requirements for AAX plug-in compatibility with Pro Tools

In addition to implementing the client-side AAX API, all Pro Tools plug-ins must:

1. Be installed to the AAX plug-ins directory
2. Use a valid file name
3. Be signed with a valid digital signature

Note that digital signatures for plug-ins are not required in Pro Tools Developer builds. However, you will need a Pro Tools Developer Build licence in order to run Pro Tools Developer. To obtain your licence, contact devauth@avid.com.

12.40.4.1 Install directories

AAX plug-ins must be installed in the system's AAX Plug-Ins directory. See [Building your plug-in installer](#) for more information about creating a plug-in installer.

Plug-ins that are uninstalled but still present on the system are placed into the "Plug-Ins (Unused)" directory, which is located next to the Plug-Ins directory.

Pro Tools will also search for a Plug-Ins directory next to the actual Pro Tools application, and this directory will be used if present. This debug feature can be useful for testing specific plug-ins.

12.40.4.2 Plug-in name and file structure

In order to be recognized by AAE, all AAX plug-in bundles must use the ".aaxplugin" file name suffix. On macOS, the plug-in bundle must use this suffix while the binary itself does not require a suffix. On Windows, the plug-in binary (DLL) must use this suffix.

The directory structure of an AAX plug-in bundle is also important. See [.aaxplugin Directory Structure](#) in the [AAX Format Specification](#) document for more information.

12.40.4.3 Digital signature

As an added security measure against digital piracy, all AAX plug-in binaries must be digitally signed in order to run in Pro Tools. This signature step does not interfere with your existing copy protection and licensing solutions - it is simply a build step that you incorporate into your plug-in before releasing the binary.

Digital signatures are generated based on the plug-in binary and act as a guarantee against binary modification. Therefore, any build steps that modify the binary, such as symbol stripping, must be performed prior to signature generation. Digital signatures apply to the full .aaxplugin bundle, so any operation that modifies the contents of the bundle will invalidate its digital signature even if the operation does not affect the plug-in binary itself. Therefore, the generation and application of a digital signature should be the last step in any release plug-in build process. The digital signature requirement applies to Beta and Release software. This requirement does not apply to Development builds of Pro Tools or to other developer tools which can load unsigned binaries.

Note

You will need a Pro Tools Developer Build licence in order to run the Pro Tools developer builds. To obtain your licence, see the Obtain a Pro Tools Developer iLok license (and an iLok) section in the Getting Started with AAX guide.

If you are having problems with digitally signing your plug-ins see the [Plug-In Fails to Load in Shipping Pro Tools](#) section in the [Troubleshooting](#) guide.

Requesting the digital signing toolkit

The AAX digital signatures required by Pro Tools are generated using digital signing tools licensed from PACE Anti-Piracy, Inc., which acts as the certificate authority for all AAX digital signatures. To request access to these tools, send an e-mail to audiosdk@avid.com with "Pace Tools Request" in the subject. Include the following information in your request:

- Company name
- Admin full name

- Email
- Telephone number
- iLok username

Once your request has been approved you will be contacted by PACE with further instructions for acquiring and using the digital signature toolkit.

What you will need

The digital signing toolkit which you will receive as an AAX developer will require a physical iLok USB key. You will also need a registered iLok user account which will be used when applying the digital signature. If your build toolchain requires hardware-free signing then you can contact PACE regarding their current offerings.

In order to successfully use the signing tools you should be familiar with the latest Gatekeeper and `codesign` (for Mac) and Authenticode (for Windows) digital signature schemes.

Although it is possible to use self-signed certificates for AAX digital signatures, before making your AAX plug-ins commercially available it is recommended that you acquire an Apple-issued Application Developer ID for Gatekeeper and an "Extended Verification" (EV) Authenticode certificate from a Microsoft approved certificate authority.

See the Getting Started Guide in the PACE digital signing toolkit for more information about using these tools.

Signature requirements in Pro Tools

Host Compatibility Notes Pro Tools requires PACE Eden digital signatures for AAX plug-ins.

Pro Tools and higher use the Eden toolset. This toolset integrates fully with platform-specific signatures, so you only need to do one post-build step using the Eden digital signing tools to sign your plug-in with both the Eden signature and the relevant Apple GateKeeper or Microsoft Authenticode signature. For more information, see the Eden digital signature toolkit documentation.

Pro Tools and higher will only accept the Eden signature; AAX plug-ins signed by earlier generations of PACE digital signing tools will not load in Pro Tools.

Optional Signature for Pro Tools AAX DSP binaries

Binary-level encryption can be added to AAX DSP algorithms. Please note that this is *NOT* a requirement of AAX DSP plug-ins, and serves only as an additional security measure to protect an algorithm's DLL.

For more information about signing AAX plug-ins for use with Pro Tools, please contact PACE.

Automatic signature application by PACE tools

If you already protect your plug-ins using one of the anti-piracy technologies available from PACE then you may not need to perform any additional action:

- you are wrapping using PACE InterLok MasterMaker, your binaries will be automatically signed.
- If you are using Fusion Hybrid without wrapping with MasterMaker, please carefully review the "Adding digital signature checks" section of the Fusion Hybrid manual.
- If you are using PACE APIs (like PACE Interface or CDRM) without InterLok wrapping, please see either the latest PACESigTool read me or the Fusion Hybrid manual for additional details regarding digital signing.

12.40.5 Audio Engine Behavior and Features

Pro Tools hosts AAX plug-ins using the *Avid Audio Engine* (AAE). AAE implements all host-side AAX interfaces such as [AAX_IController](#) and the [AAX_ICollection](#).

12.40.5.1 Plug-in loading and AAE initialization

When Pro Tools launches, it immediately begins loading AAE. AAE searches the system for valid AAX plug-ins, checks each plug-in's digital signature, and loads, initializes and catalogs any valid plug-in modules that it happens to find.

This initialization is performed via the plug-in's Describe implementation; once AAE loads a plug-in binary, it calls the plug-in's Describe method to retrieve (and cache) the basic configuration of the plug-in. AAE then hands this information back to Pro Tools so that Pro Tools knows what plug-ins are available and what their basic properties are. Once a complete list of plug-in descriptions has been generated, AAE can construct any plug-in's individual modules and manage its algorithm.

12.40.5.1.1 Plug-in configuration cacheing AAE is pretty smart, and it knows during initialization if anything has changed within the AAE Plug-Ins folder since the last time it was run. If nothing has changed, AAE relies on plug-in descriptions that it cached during the previous launch to speed through the plug-in loading process. If, however, any plug-ins have been added, removed, or updated since the last launch, AAE loads and re-caches the description for every installed plug-in.

Note

We recommend that you always enable the `AlwaysRebuildCache` DigiOption during plug-in development. See [Using DigiOptions](#) for more information.

12.40.5.2 Plug-in initialization

When a new plug-in instance is created in Pro Tools, AAE performs the following steps:

1. The plug-in's data model component is loaded
2. The default state of the plug-in is set (see [Default plug-in settings](#))
3. The plug-in's GUI and other host modules are loaded
4. The plug-in's algorithm private data state is initialized
5. The plug-in's algorithm is loaded and initialized
6. The plug-in's algorithm processing is initiated

12.40.5.3 Run-time processing behavior

The audio engine in Pro Tools includes some advanced real-time processing features that are not present in earlier versions of Pro Tools:

- When certain tracks with plug-ins have been silent for a period of time or Pro Tools is not in playback, those plug-in instances are automatically deactivated to reduce processing load on the host processor
- In certain situations such as playback or offline bounce where low latency is not required, Pro Tools may call AAX Native plug-ins with a larger buffer than normal.

This latter behavior is possible due to the fact that AAE uses two latency domains for plug-ins: a high-latency domain that operates over large block sizes and a low-latency domain that operates over small block sizes. Since processing at higher block sizes is generally more efficient, plug-in instances that are running in the high-latency domain generally consume less CPU cycles for their processing than instances that are running in the low-latency domain.

Pro Tools may swap plug-in instances back and forth between these two domains at run-time and uses a set of rules designed to optimize the system's CPU resources while at the same time providing the best and most responsive user experience in every situation. These rules are different depending on whether the system is using an HDX card as its playback engine.

Here are some of the specific rules that are followed by the current versions of Pro Tools at the time of this writing. These rules are subject to change from release to release:

- (*HDX classic and Native*) If there is any live audio or MIDI feeding into the plug-in's track and if the track is sending audio to an active output then all plug-in instances on the track will be run at low latency.
- (*HDX classic only*) If any AAX DSP plug-in instances are present in the signal path that feeds an AAX Native plug-in instance then the AAX Native plug-in will be run at low latency.
- (*HDX classic only*) Any AAX Native plug-in instance on an AUX track will be run at low latency.

For a full list of compatibility and feature differences between different AAX plug-in hosts, see [Host Support](#).

12.40.5.3.1 Deterministic Plug-in Automation Native and DSP plug-ins will receive automation changes in a deterministic manner. Each time the transport is played, automation events will be delivered to the plug-in for processing at the same moment on the timeline. Note that this does not mean automation is sample-accurate with respect to where the automation breakpoints are placed in the timeline, but rather that the timing will be the same between transport runs.

12.40.5.3.2 Deprecated and Related Plug-in Lists To help ease the transition for users to 64-bit, Pro Tools includes support for deprecated and related plug-in types. This allows you to associate any legacy plug-ins with new AAX plug-in types so that Pro Tools can automatically convert sessions with older plug-ins to the new types.

12.40.5.3.3 Offline Bounce

12.40.5.3.4 The Hybrid Engine and AAX DSP Pro Tools HDX and Pro Tools Carbon systems support the Hybrid Engine, which optimizes system latency by splitting the Pro Tools mix topology between native and DSP processors. When the Hybrid Engine is in use, Pro Tools tracks are configured as either DSP Mode or Native Mode. All [AAX](#) effect instances on a DSP Mode track are switched to their [AAX DSP](#) type. Plug-ins that do not support [AAX DSP](#) are de-activated while the track is in DSP Mode. Pro Tools HDX also supports a Classic mode. In this mode, the user chooses whether each plug-in instance will be [AAX Native](#) or [AAX DSP](#).

Pro Tools supports faster-than-real-time offline bounce for all sessions. All plug-ins with AAX Native types are supported. For AAX DSP plug-ins, the offline bounce process will temporarily convert those to their corresponding AAX Native types to complete the bounce. Because offline bounce is faster-than-real-time, audio processing callbacks will occur as fast as the algorithm will allow for. For this and other reasons, your algorithms should never depend on wall-clock time for features such as LFO, delay time, etc. Instead, all algorithms should always base time calculations on sample time so that the output will still be correct even if the algorithm is being called from an offline bounce.

12.40.5.3.5 AAX Hybrid Plug-ins Pro Tools also supports [AAX Hybrid](#) plug-ins, which can have both a Native and a DSP algorithm processing component. Audio-rate data can also be shared between the two processing components, which allows you to split your algorithm up into low-latency and high-latency contexts for better efficiency and to enable plug-ins such as convolution reverbs, spectrum analyzers, and other similar architectures.

Note

AAX Hybrid is only supported by Pro Tools HDX when running in Classic mode. See [PT-257213](#) for more information.

12.40.6 Basic plug-in operation

12.40.6.1 Configuration management

Each Effect in an AAX plug-in may contain multiple configurations. Pro Tools automatically determines the appropriate plug-in configuration for each Effect insert at run time based on the insert's required sample rate, channel width, and processing mode (Native or DSP.) Under some circumstances, the configuration requirements for an Effect insert may change at run-time. Here are some examples:

- When a width-changing plug-in (e.g. mono-to-stereo) is instantiated on a track then all of the following inserts must be converted to the new stem format
- When a user imports session data between sessions at different sample rates then all of the imported plug-ins must be converted from the old sample rate to the new sample rate
- When a user opens a session that contains deprecated effects, they must be replaced by the corresponding installed effects

Whenever a new configuration is required, Pro Tools automatically determines whether one is available that meets the new requirements and, if it is, swaps in a new plug-in instance using a copy of the previous configuration's settings.

In order for Pro Tools to deterministically select the appropriate Effect configuration to load in any given scenario, each of the configurations that are registered in the Effect must be described with distinct and mutually exclusive compatibility requirements.

12.40.6.2 Plug-in activation and deactivation

In Pro Tools, real-time plug-in inserts can be either active or inactive. Inactive plug-ins are not instantiated and are entirely removed from the processing chain, though they are still saved with the session and maintain a placeholder in their track's insert list for easy activation at a later point.

Active plug-ins may be de-activated manually by the user or automatically by Pro Tools. Plug-ins may be loaded as inactive when a plug-in that has been saved in a session has been uninstalled and is no longer available, when a required plug-in configuration is not available, or at any other time when a particular plug-in instance cannot be loaded.

12.40.6.3 Plug-in bypass

AAX plug-ins must implement a Master Bypass parameter, which is controlled via the "Bypass" button in the Pro Tools plug-in window header. While bypassed, the plug-in must not apply any processing to the audio that is passed to it (except delay, see below.) The plug-in may choose to smoothly transition into and out of bypass however it chooses.

Any algorithmic delay that a plug-in incurs during normal operation must be maintained by the plug-in during bypass. For more information about this requirement, see [Automatic Delay Compensation](#).

12.40.6.4 Presets and settings management

Pro Tools includes a plug-in preset management system that can be accessed from the plug-in window header. With this system, users can save plug-in settings to disk and load the settings later to restore the plug-in's configuration.

Preset files can be bundled with an AAX plug-in to demonstrate a variety of uses for the plug-in or as recommended settings for different situations, and, as a plug-in developer, you are encouraged to provide a large selection of pre-configured presets along with your AAX plug-ins. See [Create factory presets](#) for more information about bundling presets with your plug-in.

Aside from user preset management, there are many cases when the state of a plug-in must be captured or restored by AAE. For example, AAE must restore plug-in settings when a session is loaded and when converting a plug-in between different configurations.

12.40.6.4.1 The plug-in preset menu Plug-in presets are available to the user via the Plug-In Settings menu in the Pro Tools plug-in window header. This menu supports nesting presets into sub-folders, which provides a convenient way to categorize and organize large sets of presets. In addition, users may save custom presets and add these custom presets to the menu.

Figure 2: The Plug-In Settings menu in the Pro Tools plug-in window header

The preset menu in the Pro Tools plug-in header is built from the following two directories:

- *Session file*/../Plug-In Settings
- *User Library root*/Plug-In Settings

The default setting for the User Library root directory is ~/Documents/Pro Tools on macOS, but the user can change this setting in the Pro Tools preferences.

12.40.6.4.2 Factory presets Pro Tools supports automatic installation of plug-in presets. AAX plug-ins should include a set of presets in the following directory within the .aaxplugin:

- *MyPlugIn.aaxplugin/Contents/Factory Presets/MyPlugInPackage/*

Where *MyPlugInPackage* is the plug-in's longest [Package Name](#) with 16 characters or fewer.

On Pro Tools launch, all installed AAX plug-in settings are copied from the .aaxplugin bundles' "Factory Presets" folders into the User Library directory (see [above](#).)

Note

Since the User Library root directory is a customizable setting, you should never install presets directly onto a user's system. If you require a central repository of settings on the system that is under control of your installer then you should handle these settings as external resources. You can use custom settings chunks in the plug-in's "Factory Presets" .tfx files to redirect your plug-in to read the appropriate installed resources.

12.40.6.4.3 Default plug-in settings When the first instance of an effect is made active in a session, Pro Tools queries the instance's state and stores this data as the effect's "factory default" preset. This preset is cached by Pro Tools and will be set on each subsequent instance of the plug-in with the same configuration

The plug-in's factory default settings are stored on disk in a temporary file location that is specific to the user. Pro Tools looks for the factory default settings file for a plug-in each time an instance of the plug-in goes from an inactive state to an active state, including when the instance is first created. If there is no factory default settings file on disk then Pro Tools will create it using the plug-in's current settings.

All factory default settings files are deleted during the Pro Tools shutdown procedure. Therefore, under normal operation, these files will be refreshed with each launch of Pro Tools.

Note

If the Pro Tools shutdown procedure is not completing, for example if you regularly terminate Pro Tools from a debugger, then the plug-in factory default settings files will not be deleted automatically.

When a session is loaded Pro Tools will perform the following steps on each plug-in instance:

1. Instantiate the plug-in and create a [AAX_IEffectParameters](#) object
2. Look for the cached factory default settings file in the file system
3. If the factory default settings file is not found, query the plug-in for its current settings and create the factory default settings file using these settings
4. Set the instance's default settings based on the settings stored in the cached factory default file
5. Send the instance a [AAX_eNotificationEvent_SessionBeingOpened](#) notification
6. Set the saved settings from the session

12.40.6.4.4 The Compare Light The plug-in window header in Pro Tools includes a "Compare" button, the Compare Light control. This control allows the user to compare the current state of the plug-in with the last preset that was loaded, or the plug-in's default settings if no other preset has yet been loaded.

Pro Tools polls each displayed plug-in periodically to determine whether or not its state matches the currently loaded preset. While the state matches, the Compare Light is inactive and unlit. As soon as the plug-in's state differs from the preset, the Compare Light becomes active and is highlighted.

When the Compare Light is active, the user may click on it to cache the current plug-in settings and temporarily swap in the last preset that was loaded. Clicking on the Compare Light a second time will restore the cached plug-in settings.

The specific operation of the Compare Light is determined by the plug-in's implementation of [AAX_IEffectParameters](#). To determine the correct state for a plug-in's compare light, Pro Tools makes regular calls to [AAX_IEffectParameters::GetNumberOfChannels](#) from a callback timer. If this method's `aValueP` parameter has changed since the last time the plug-in was queried then Pro Tools proceeds to call [CompareActiveChunk\(\)](#). If [CompareActiveChunk\(\)](#) returns with `isEqual==false` then the Compare Light will be lit, otherwise the light will be dimmed.

12.40.6.4.5 Basic chunk handling All of these situations use the same basic settings management infrastructure in Pro Tools, which uses the "chunk" API of [AAX_IEffectParameters](#) to retrieve arbitrary blocks of data from the plug-in (to retrieve a preset) and send the same block back to the plug-in (to set a preset.)

When retrieving a preset from a plug-in, Pro Tools first asks for the size of the plug-in's settings chunk(s). Pro Tools then provides the plug-in with a pre-allocated buffer of memory into which the plug-in may store its settings information using any format that it chooses.

When Pro Tools needs to restore the plug-in to this preset state, it sends a copy of this data back to the plug-in. The plug-in must interpret this data and set its internal state to match the preset.

12.40.6.5 Modifier key behavior

In order for users to have a consistent experience, all AAX plug-ins should provide standard modifier key behaviors as described in this section. These operations are demonstrated by all Avid plug-ins in Pro Tools, and you can experiment with Avid's AAX plug-ins to demonstrate the correct plug-in modifier key behavior.

The following modifier key combinations must be handled explicitly by the plug-in:

macOS Keys	Windows Keys	Expected Behavior
Command-click	Control-click	Adjust the parameter's value with fine control, for continuous controller widgets
Option-click	Alt-click	Return the parameter's value to default*
Shift-click	Shift-click	Link parameters across all channels, if applicable
*Set-to-default may also be handled directly by the host, depending on the host version (see below).		

In addition to these events, there are also specific behaviors which Pro Tools and other AAX hosts provide for certain key combinations in plug-in GUIs. For example, Pro Tools provides the following modifier key behavior overrides:

macOS Keys	Windows Keys	Expected Behavior
Command-Control-click Command-Right click	Control-Start-click Control-Right click	Show parameter automation lane in the Pro Tools Edit Window, if automation is enabled for the parameter
Command-Option-Control-click Command-Option-Right click	Control-Alt-Start-click Control-Alt-Right click	Activate pop-up menu for automation

Other AAX plug-in hosts implement different host-managed behavior for modifier key combinations, and additional host-managed key combinations may be added to any AAX host in the future. For example, Pro Tools adds host-managed support for setting plug-in parameters to their default values.

In order to allow the AAX host to handle these operations, a plug-in must always call the handler methods in [AAX_IViewContainer](#) before handling any mouse events in its own GUI. It is important to call these methods for *all* mouse events, in case additional handlers are added to future versions of the host or the plug-in is run in a new AAX host with a different set of handled modifier key combinations.

See the [AAX_IViewContainer](#) class documentation for more information about passing mouse events to the AAX host.

12.40.7 Optional plug-in features

Pro Tools plug-ins offer users a rich set of integrated features. To make sure your plug-ins integrate into users' expected Pro Tools workflows, where applicable you should implement all of the features presented in this chapter.

For more information about any of these features in Pro Tools, see the latest Pro Tools Reference Guide.

12.40.7.1 Audio management features

12.40.7.1.1 Sidechain input If applicable, plug-ins may choose to enable [sidechain inputs](#). If a sidechain is enabled, a menu is added to the plug-in's header that allows the user to choose an interface or bus as the sidechain, or "key input". For AudioSuite, the user can only use an existing audio track as the sidechain input. Once enabled, the plug-in will be able to access sidechain input just like any other input signal.

12.40.7.1.2 Auxiliary Output Stems Pro Tools has the capability to show and route multiple "auxiliary" outputs from a plug-in to other tracks. These are known as [Auxiliary Output Stems](#) (AOS), a stem referring to one set of outputs. A stereo stem contains two outputs, left and right, and a mono stem contains one output. The outputs will appear in the input assignment pop-up menu of each track under the category "plug-in".

Some notes regarding this feature:

- Only mono and stereo stems are available as auxiliary outputs.
- The aux outputs cannot be added and removed from the system dynamically though they can be made inactive by the user. The total number of aux outputs, stem types, names, paths, and ordering are defined only once by the plug-in.
- Plug-in aux outputs are not available from the sidechain input popup menu in other plug-ins. Users will not see the "plug-in" submenu when clicking on a plug-in sidechain popup.
- There cannot be any multi-mono multi-output plug-ins. If a mono plug-in instance offers multiple outputs it cannot support multi-mono.

If a plug-in is going to utilize the AOS feature, it will be responsible for a few details that are summarized below:

- **Aux Output Paths**

The plug-in is responsible for the definition of valid aux output paths. This definition includes the total number of outputs, the desired order of stereo and mono paths. Pro Tools will query each plug-in for available valid paths and populate its track input selector popup menus accordingly.

- **Aux Output Path Order**

The plug-in is responsible for specifying the type and name of each of its aux output paths. A plug-in decides whether the aux outputs are all stereo, all mono, "X" stereo outputs followed by "Y" mono outputs, or some other combination. Pro Tools lists each output in the order given by the plug-in. If mono and stereo paths are interleaved the input popup menu of the mono tracks keeps that order and breaks the stereo paths into their respective left and right sides using ".L" and ".R" suffixes.

- **Aux Output Names**

A plug-in is responsible for giving meaningful names to aux outputs. Names are only defined once, so they will stick. At the very least, individual outputs should be labeled "Output xx", where "xx" is the aux output number as it is defined in the plug-in. The output name should also include the words "mono" and "stereo" to support when users are looking for an output with a specific stem format.

- **Aux Output Numbering**

The plug-in is responsible for defining the lowest available aux output number. Plug-ins should base this number on the width of the plug-in's main outputs. For example, when using a stereo instance of a sampler the first aux output should be #3, when using a 5.1 instance of the sampler the first aux output should be #7, etc. This is to keep the numbering scheme inside of the plug-in and in Pro Tools consistent. From the perspective of Pro Tools, plug-ins typically enumerate all available outputs and do not differentiate between main and aux outputs. The first "N" outputs are used for the main outputs, and all the remaining outputs are available for aux output paths.

- **Separate Multi-Output Plug-in Process Type**

Plug-in developers are encouraged to offer both "regular" and "multi-output" versions/types of any multi-output capable plug-in. We strongly suggest this to conserve resources and to keep the user's workspace as uncluttered as possible. Users can choose to use the regular version/type for plug-ins they don't need aux outputs for. Multi-output versions can be created as separate process types so that there need not be separate binaries. Such additional process types will be listed in the plug-in menu next to their regular version siblings. They should be nominally distinguished by appending phrases like "multi-output" to the plug-in name, for example.

Note

When moving sessions between different PT systems, multi-output process types will NOT be automatically converted to regular process types if multi-output types are not available.

- **No Multi-Mono Implementations**

A plug-in is responsible for not having multi-mono enabled if it utilizes auxiliary outputs stems. Auxiliary output stems will not work in multi-mono enabled plug-ins. Multi-mono is automatically disabled for AOS in the Effect Layer.

12.40.7.1.3 External metering and internal clip Pro Tools may use the meter values reported by a plug-in for display on attached control surfaces and other external plug-in views. In general, the behavior of a plug-in meter on these devices will depend on the meter's properties as registered in Describe. The meter behavior may also depend on the plug-in's registered category. See [Plug-in meters](#) for more information about how to register your plug-in's meters.

- **Gain reduction metering**

Pro Tools supports gain reduction metering and will display an inverted gain reduction meter next to each plug-in insert and also next to the track's main meter in the Pro Tools Mix and Edit windows.

All registered plug-in gain reduction meters are used by the Pro Tools gain-reduction metering UI. The plug-in gain reduction meters in the Pro Tools Mix and Edit windows will combine metering data for all gain-reduction meters of the same type ([Compressor/Limiter](#) or [Expander/Gate](#)) in the plug-in:

- For plug-ins with multiple gain-reduction meters of the same type, the minimum (most gain-reduced) meter value for the current buffer will be used

- For multi-mono plug-ins, the minimum meter value across all of the per-channel mono instances will be used

Pro Tools can be set up to display [Compressor/Limiter](#), [Expander/Gate](#), or both types of metering data in these displays via Preferences > Metering > Display > Gain Reduction Meter Type. If both types are used, the displayed meter level is simply the sum of the selected values for each type.

The track gain-reduction meter displays the sum of all the track's inserts' gain reductions, using the same rules as above.

- **Plug-in internal clipping**

The plug-in header has a clip light that indicates that the plug-in has reported that it has clipped somewhere internally. It is up to the plug-in itself to set and clear its clip indicators as needed. Additionally, plug-ins that have clipped internally will appear in red on the insert, even if the plug-in window is not open. This allows users to see at a glance where clipping has occurred in their mix.

12.40.7.1.4 Automatic Delay Compensation Automatic Delay Compensation maintains time-alignment between tracks that have plug-ins with differing algorithmic delays, tracks with different mixing paths, tracks that are split off and recombined within the mixer, and tracks with hardware inserts. To maintain time alignment, Pro Tools adds the exact amount of delay to each track necessary to make that particular track's delay equal to the delay of the track that has the longest delay.

In order to be compensated correctly, AAX plug-ins must report any algorithmic delay that they incur. This delay may be reported in the plug-in's description, and may also be changed at run-time.

Automatic Delay Compensation Notes

- Currently, Pro Tools will not update its delay compensation settings while Pro Tools is in playback, so a plug-in that dynamically changes its delay settings at run-time should either prevent any algorithmic delay updates during playback or give a visual indication to the user when the delay that it applies and the delay that Pro Tools is compensating for different delay settings.
- Pro Tools does not update delay compensation settings when plug-ins go into and out of bypass, and does not automatically maintain bypass audio buffers for delayed plug-ins. It is therefore required that all plug-ins incur the same amount of delay when bypassed as during normal operation.
- Automatic Delay Compensation is not applied during offline (AudioSuite) processing for plug-ins which use the [Host Processor](#) interface. If your Host Processor plug-in incurs algorithmic delay then you must incorporate audio lookahead via the Host Processor interface's random timeline access API.
- Given the many routing possibilities in Pro Tools, the Automatic Delay Compensation feature involves some subtleties that may not be immediately apparent or intuitive. For more information about this feature, we strongly recommend that you review the relevant chapters in the Pro Tools Reference Guide.

12.40.7.2 Plug-in categories

The plug-in menus in Pro Tools are hierarchical and by default are organized by category. These general categories represent common plug-in functions like EQ, dynamics, reverb, etc. Plug-ins may report one or more of these categories in order to be placed into the proper menu. For a complete list of plug-in categories available, refer to the [AAX_EPlugInCategory](#) enum.

Some features, such as control surface center-section mappings, are only available to plug-ins that report a particular category, so it is important for plug-ins to report the correct set of categories.

12.40.7.3 Advanced non-real-time processing

AudioSuite processing allows AAX plug-in to operate on audio in a non-real-time manner. AudioSuite plug-ins will appear in the AudioSuite menu in Pro Tools. By default, any AAX-Native plug-in will appear in the menu as long as an [AAX_eProperty_PluginID_AudioSuite](#) property is defined alongside the corresponding [AAX_eProperty_PluginID_Native](#) ID. However, to make use of extended AudioSuite features such as non-real-time sample access, the Analysis pass, a separate entry method subclassed from the [AAX_CHostProcessor](#) implementation in the SDK should be used.

12.40.7.3.1 AudioSuite processing modes Pro Tools includes a number of different AudioSuite processing modes, each of which changes the precise behavior of an AudioSuite processing event.

Output modes

- *Overwrite files* Output audio destructively overwrites the selected audio files on disk
- *Create individual files* Individual new files are created for each processed clip
- *Create continuous file* A single new file is created with data from the full processing pass

Input modes

- *Clip by clip*
- *Entire selection*

The plug-in may optionally disable the "clip by clip" processing mode if continuous input data is required, by using the property [AAX_eProperty_ContinuousOnly](#).

Channel modes

- *Mono mode* Each selected channel is processed as an individual mono audio stream
- *Multi-input mode* Selected channels are sent to the plug-in in multi-channel streams

Multi-input mode is only valid with the "entire selection" input mode, since the "clip by clip" input mode requires that each clip be processed individually as a standalone audio channel.

The plug-in may optionally disable "mono mode" processing if its algorithm is only valid for multi-channel input, by defining the [AAX_eProperty_MultiInputModeOnly](#) property.

12.40.7.3.2 Analysis AudioSuite plug-ins support and optional Analysis pass, which allows a plug-in to access the incoming audio before the actual Render pass starts. When Analysis is defined with either [AAX_eProperty_OptionalAnalysis](#) or [AAX_eProperty_RequiresAnalysis](#), an Analyze button will appear in the plug-in footer in the GUI.

An analysis pass is useful for collecting pitch, spectrum, loudness, noise threshold, or other data that will help the user set up parameters based on the audio content being processed.

12.40.7.3.3 Reverse A "reverse" feature is available for [Reverb](#) and [Delay](#) AudioSuite plug-ins. This effect will reverse the source audio, apply the AudioSuite plug-in processing, and re-reverse the source audio back to its original orientation, thereby applying the AudioSuite effect in reverse.

Reverse replaces the Analysis pass in the Pro Tools UI, so AudioSuite plug-ins in these categories do not receive a user-triggered analysis pass.

12.40.7.3.4 Random-access and non-linear processing The generation of output samples by an AudioSuite Process must occur linearly and incrementally; however, the source of input samples may optionally be randomly accessed from the entire selected track. This enables many advanced processing capabilities such as whole-file analysis, audio reverse effects, and timeline-level modifications such as expanding, contracting, or shifting the processed region.

In order to prevent invalid data from being randomly accessed, the "overwrite files" processing mode is disabled for plug-ins that use random-access processing.

12.40.7.3.5 AudioSuite Handles By default, when processing audio segments with an AudioSuite plug-in, Pro Tools will also process an extra region before and after the selected audio. These extra regions will be trimmed out of the selected.

The reason for this feature is so that the user has some room to expand the resulting audio clip (for fades or other reasons). However, certain AudioSuite plug-ins will operate more intuitively if these handles are not processed (such as delay, reverb, loudness normalization, and other plug-in types). To disable extended handle processing, set the [AAX_eProperty_DisableHandles](#) property to true.

12.40.7.3.6 Extended features AAX-AudioSuite plug-ins also have several other optional features including custom progress dialogs, reverse mode, and side-chains. For a complete reference of supported AudioSuite-related properties, refer to the properties between [AAX_eProperty_AudiosuitePropsBase](#) and [AAX_eProperty_MaxASProp](#) found in [Interfaces\AAX_Properties.h](#).

12.40.8 Using the Pro Tools Scripting SDK with AAX

The Pro Tools Scripting SDK provides a way to control various aspects of Pro Tools. AAX plug-ins may incorporate the Pro Tools Scripting SDK in order to control Pro Tools in ways that are not possible through AAX alone.

The Pro Tools Scripting SDK requires a PTSL connection to be established with Pro Tools. When establishing a PTSL connection from within an AAX plug-in, the connection request must be made using a non-main application thread. Spawn a new thread from within your AAX plug-in to perform all PTSL connection requests and Pro Tools Scripting SDK commands.

12.40.9 Debugging AAX plug-ins

12.40.9.1 Debugging within Pro Tools

Shipping versions of Pro Tools do not support attaching a debugger. This is to prevent malicious users from compromising Pro Tools security as well as the security of third-party plug-ins.

As an AAX plug-in developer, you are granted access to debuggable "developer build" versions of Pro Tools to help your development efforts. Some Pro Tools developer builds are feature-limited; for example, developer builds of Pro Tools do not allow saving or exporting sessions.

The easiest way to debug plug-ins within Pro Tools is to start up Pro Tools, open a session, attach your debugger, and then instantiate your plug-in. This order seems to work the best for most users. If you need to debug the initial host query of your plug-in at Pro Tools start, it is possible to launch Pro Tools from within your debugger. However, this method is sometimes known to cause problems with certain debuggers.

AAX plug-in developers are also able to download pre-release and beta versions of upcoming Pro Tools releases. For now, these pre-release versions are not debuggable, but that is expected to change in the future as we work to make a unified debuggable pre-release installer available for upcoming Pro Tools versions.

Both debuggable and pre-release versions are available for download as part of the AAX SDK Toolkit on the [My Toolkits and Downloads](#) page at [avid.com](#). In order to use developer and pre-release builds, you will need special licences which you must request from devauth@avid.com.

12.40.9.2 DigiShell

DigiShell is a software tool that provides a general framework for running tests on Avid audio hardware. As a command-line application, DigiShell may be driven as part of a standard, automated test suite for maximum test coverage. The latest DigiShell tools may be downloaded as part of the AAX SDK Toolkit on the [My Toolkits and Downloads](#) page at [avid.com](#).

More information about DSH in general and about loading and testing plug-ins in DSH can be found in [DSH Guide](#).

12.40.9.3 DigiTrace

All Avid AAX hosts provide tracing functionality based on Avid's DigiTrace tool. DigiTrace is a library that provides high-performance logging and tracing capabilities to Pro Tools and its components, including plug-ins. More information about DigiTrace can be found on Avid's audio developer website.

To enable trace logging for AAX plug-ins, use the `AAX_TRACE` macro defined in `AAX_Assert.h`. A separate macro, `AAX_ASSERT`, is also available to signal run-time errors. These macros are both cross-platform and will function whether the algorithm is running on the TI or on the host.

For more information about DigiTrace, see [DigiTrace Guide](#).

12.40.9.3.1 Tracing requirements The `AAX_ASSERT` and `AAX_TRACE` macros are debug-only and will not provide tracing output from release builds of your plug-in. `AAX_TRACE_RELEASE` may be used for tracing in both debug and release configurations. These macros require that the `DTF_AAXPLUGINS` facility to your DigiTrace configuration file. You can toggle this facility to enable or disable AAX algorithm-level tracing. For details on setting up tracing on AAX TI plug-ins, please refer to the [HDX DSP Guide](#).

12.40.10 Troubleshooting common AAX plug-in failures

Pro Tools presents a "Move Unauthorized Plug-ins" dialog after attempting to launch with my plug-ins installed, and the plug-ins do not appear in the Pro Tools insert menus

- This error indicates that Pro Tools was not able to load the plug-in binary for some reason. The error indicates a copy protection failure since that is by far the most common reason for users to encounter this kind of error in released plug-ins, but any other error that prevents the plug-in DLL from loading in Pro Tools may also cause this error message.

This error does *not* indicate a failure when checking the plug-in's digital signature. A digital signature failure would generate a different error message that would specifically mention the plug-in's signature.

The `DTF_AAXHOST` [DigiTrace](#) facility provides additional information about AAX plug-in load errors.

One common cause of DLL loading failures is a failure to dynamically link to other required libraries. In this case, the `DTF_AAXHOST` tracing will indicate something like the following:

```
- AAXH_CEffectFactory::ParseDLL - exception thrown(The specified module could not be found. (126) while
```

This exception indicates that some DLL upon which the plug-in depends is not present in the system. This is most commonly due to dynamic linking to CRT libs, but it could also be caused by any other DLL dependency.

12.40.11 Using DigiOptions

DigiOptions provide a way to override the standard Pro Tools configuration. These options are designed to assist with testing and development of Pro Tools, and they are often useful during plug-in development as well.

To configure DigiOptions, place a plain-text file named DigiOptionsFile.txt next to the Pro Tools application. On macOS, place this file next to the Pro Tools.app bundle and on Windows place it next to the Pro Tools executable.

A red notice will appear in the Pro Tools splash screen and in the application About Box indicating when DigiOptions are enabled in the build.

Figure 3: DigiOption activation notice in the Pro Tools splash screen.

Note

If suffixes are hidden on your OS then be careful that you do not accidentally give the file an incorrect name such as DigiOptionsFile.txt.txt.

12.40.11.1 Useful DigiOptions

Note

Support for these options may vary from release to release

- `AlwaysRebuildCache 1`

This option forces Pro Tools to re-load all installed plug-ins each time the application is launched. This can be very useful during development, since some plug-in updates during development will not result in an updated plug-in binary or an updated modification date for the top-level .aaxplugin bundle.

Note

If you have changed your plug-in's ID values during development and if you encounter AAE error -20038 when your plug-in is loaded then Pro Tools may be using a cache of the outdated plug-in ID. Use the AlwaysRebuildCache DigiOption or launch Pro Tools once without your plug-in installed to clear this state.

- `NeverUnloadPlugInBundles <int>`

Enable plug-in bundle unloading. The default value for this option is 0. In order to test your plug-ins and make sure that they operate correctly this option must be set to 0.

- `LogInterruptDataEveryNSeconds <int>`
`LogInterruptDataEveryNSeconds_HL <int>`

These options enable regular DigiTrace performance logging from the low-latency and high-latency real-time audio render threads in the Avid Audio Engine. For example, `LogInterruptDataEveryNSeconds↔_HL 2` would trigger a performance log for the high-latency render thread every two seconds. For more information about performance logging in AAE see [Real-time AAE performance logging with DigiTrace](#).

- `PauseDuringLaunchToAttachDebugger 1`

- `DisplayHostPlugInLatency 1`

Display information about the plug-in's processing domain and dynamic processing status in the Pro Tools plug-in window header.

Figure 4: DisplayHostPlugInLatency info in plug-in header.

- `TestGetCurveData <0, 1, 2>`

Display the plug-in's curve data as a plot in the Pro Tools plug-in window header. Possible values are:

0. Off
1. [EQ curve](#)
2. [Gain reduction curve](#)

Warning

The implementations of this test curve and the actual curve data shown on Avid control surfaces are different. In particular, the display in Pro Tools is updated regularly at idle time, whereas the curve on a control surface is only updated when certain parameter changes occur (see [PTSW-195316 / PT-218485](#)). The graph point interpolation, range, and sample points are also not exactly equivalent to the values used on an actual control surface, so you should not expect the curve shown in the debug view in Pro Tools to exactly match what would appear to users. After performing early prototyping of your curve data using the `TestGetCurveData` DigiOption you are strongly encouraged to use either the S6 Surfulator software or the [Pro Tools | Control](#) app to test and refine the plug-in's curve data in a real-world environment.

For more information about graph curves see [EQ and Dynamics Curve Displays](#).

Figure 5: `TestGetCurveData` EQ graph in plug-in header.

- `RenderMissingFilesAsBlank 1`

This option may be used to automatically render test tones into audio clips with missing source media. The test tones are rendered with different frequencies and magnitudes. This option can be useful when troubleshooting using a session file provided by an end user, since session-specific issues are rarely dependent on the source media, but may depend on there being some signal present in the session. This option requires that the Avid Signal Generator plug-in is installed.

- `44100_Rate <int>`
`48000_Rate <int>`

Set the new base sample rate for each set of sample rate multiples. Can be useful for simulating sample rate pull-up by up to +5% (e.g. `44100_Rate 45000` in `DigiOptions.txt`.) The `44100_Rate` option affects 44100, 88200, and 176400 Hz rates, while the `48000_Rate` option affects 48000, 96000, and 192000 Hz rates.

- `EnablePlugInHotSwap 1`

Note

This option is currently non-functional for AAX plug-ins Pro Tools. See [PTSW-188653 / PT-218451](#)

This option will allow Pro Tools to recognize changes to your plug-in during run-time. This allows you to re-compile and load your updated plug-in without re-launching Pro Tools. The following conditions must be true in order to enable hot-swapping between versions of your plug-in:

- There cannot be any instances of the plug-in currently in Pro Tools.
- Both `EnablePlugInHotSwap 1` and `AlwaysRebuildCache 1` must be set.

- `WinDLLErrorMode <int>`

Set the Windows error mode during DLL loading and unloading. The value of this option will be set as the `uMode` for a call to the `SetErrorMode` Windows API during DLL loading and unloading.

This option can be useful when debugging plug-in load errors on Windows, for example errors that cause a "The following Plug-Ins failed to load because no valid authorization could be found" dialog to appear during Pro Tools launch. <DIV CLASS="SectionHead"> <TT>DisableCMNAssert 1</TT></DIV> Disable assert dialogs in Pro Tools. Use this option if your Pro Tools debugging sessions are being interrupted or terminated due to assert failures in the app. Note that Pro Tools asserts may be triggered by your plug-ins; you should always investigate any assert that you see to determine whether it is being caused by a plug-in. If you need information about any Pro Tools asserts, post a question here on the %AAX developer forums or write to us at devservices@avid.com and we will be happy to help. <DIV CLASS="SectionHead"> <TT>TestPlugInDescriptions 0</TT></DIV> Use this DigiOption to toggle the plug-in description validation check in Pro Tools developer builds. Developer builds will check plug-in description information when the plug-in is loaded and will present a warning dialog if the check fails. See \ref describe_validation section in the \ref CommonInterface_Describe page for more information. <DIV CLASS="SectionHead"> <TT>RealTimeDenormalsAreZero <int></TT></DIV> Use this DigiOption to toggle the default denormal handling policy of the AAE real-time processing threads. A value of 1 means that DAZ+FZ will be enabled for all %AAX real-time processing threads unless explicitly changed using

thread-specific primitives, while a value of 0 means that DAZ+FZ will be disabled unless explicitly changed. The default state of the DAZ+FZ flags for AAE real-time processing threads is turned on by default. <DIV CLASS="SectionHead"> <TT>DigiTraceWindow 1</TT></DIV> Enable the Console window in Pro Tools which displays the application's \ref AAX_DigiTrace_Guide "DigiTrace" output in a live stream. \xrefitem compatibility_notes 12. </DIV> <DIV CLASS="section">

12.40.12 Compatibility Notes

See [Host Support](#) and [Known Issues in Pro Tools](#) for additional details regarding Pro Tools compatibility

Collaboration diagram for Pro Tools Guide:



12.41 Media Composer Guide

Details about using AAX plug-ins in Media Composer.

12.41.1 Contents

- [About this document](#)
- [Processing modes](#)
- [Compatibility requirements](#)
- [AAX feature support in Media Composer](#)
- [Additional Information](#)

12.41.2 About this document

This guide discusses specific details related to using AAX plug-ins with Media Composer, such as loading and initialization procedures, GUI hosting, and other application-specific features.

For more information about the features, functionality, and use of Media Composer see the Media Composer user documentation.

12.41.3 Processing modes

Media Composer supports AAX plug-ins in two processing modes: AudioSuite and AAX Native

- AudioSuite plug-ins perform non-real-time, random-access, file-based processing entirely on the host CPU.
- AAX Native plug-ins perform real-time, linear, non-destructive processing entirely on the host CPU.

AAX plug-in processing in Media Composer is managed by specific Tools. Each of these Tools can be accessed using the "Tools" menu in the Media Composer application.

12.41.3.1 Non-real-time processing (AudioSuite)

Use the AudioSuite Tool to perform AudioSuite processing in Media Composer. The AudioSuite Tool applies an effect to a clip in the timeline of the record monitor.

Specific AudioSuite plug-ins appear in the Plug-In Selection menu in the AudioSuite window.

Note

Unlike Pro Tools, the effect to clip relationship is remembered along with the effect parameters used. Parameters to the effects can be changed at a later time, and at any time the effect can be re-rendered with the saved effect parameters. Therefore it is very important for AudioSuite plug-ins to maintain compatibility between instances, versions, and systems in order to function properly in Media Composer workflows. See [Preset management](#) for more information.

Media Composer supports two AudioSuite processing modes:

- Apply a plug-in to a clip in the Timeline. This method creates a rendered effect.
- Use the controls in the AudioSuite window to create a new master clip. This method lets you process more than one channel at a time and to create new media with a duration longer or shorter than the source media.

By default, the AudioSuite window displays the controls for applying a plug-in to a clip in the Timeline. When you drag a master clip into the window, the window expands to display additional parameters for working with master clips.

12.41.3.1.1 Applying an AudioSuite Plug-in to a Clip in the Timeline The following illustration shows the default layout of the AudioSuite window:

Figure 1: The AudioSuite window

To apply an AudioSuite plug-in to a clip in the Timeline:

1. Open the AudioSuite window by doing one of the following:
 - Select Tools > AudioSuite
 - If an audio tool is already open, click the Effect Mode Selector menu and select AudioSuite
2. Use the Track Selection Menu button to select the tracks that you want to modify.
 - When you select an item from this menu, the system selects or deselects the corresponding track in the Timeline

- To select multiple tracks, press the Shift key while you select additional tracks from the Track Selection menu. Plus signs (+) mark the additional tracks and indicate that the effect is applied to more than one track.
3. Click the Plug-In Selection menu, and select a plug-in
 4. Click the Activate Current Plug-In button. This opens a dialog box associated with the plug-in.

From the AudioSuite dialog box, you may make any necessary adjustments to the plug-in and Preview the effect in real-time.

- To save the effect, click OK
- To close the dialog box without saving the effect, click Cancel
- To save the effect as a template, drag the effect icon to a bin

12.41.3.1.2 AudioSuite Master Clip Mode Drag a Master Clip into the AudioSuite Tool to engage AudioSuite Master Clip Mode. This mode supports all AudioSuite effects, including those that change the width or length of the effected clip. A new Master Clip is generated for each AudioSuite processing pass applied in this mode.

In Master Clip Mode, the AudioSuite window will be expanded to display additional controls. You can also click the Display/Hide Master Clip Controls button to display or hide the additional parameters.

The following operations are available in Master Clip Mode:

- Apply AudioSuite plug-ins to more than one track at the same time. For example, a plug-in might let you process two separate tracks as a stereo pair. This enables you to use plug-ins that perform linked compression, reverb, and other effects that allow multichannel input.
- Create new media with a longer or shorter duration than the source media. This lets you use effects that perform time compression and expansion. For example, you can use a Time Compression Expansion plug-in to change the length of the audio file, or you can lengthen the file in order to add a reverb trail.
- Apply one mono AudioSuite effect to multiple inputs of a master clip in a multiple-mono fashion.

For more information about processing in Master Clip Mode, see the Media Composer user documentation.

12.41.3.1.3 Restrictions on AudioSuite processing

- Media Composer does not support width-changing AudioSuite effects except in [Master Clip Mode](#). See [Processing configurations](#) for more information about supported stem formats in Media Composer.
- AudioSuite effects that change the clip length should only be used in [Master Clip Mode](#), because consolidated sequences will not consolidate the correct media length.

12.41.3.2 Real-time processing

Use the Audio Track Effect Tool to perform real-time processing in Media Composer. Audio Track Effects appear in the Audio tab of the Effect Palette, as well as in the menus of the Audio Track inserts in the Audio Mixer Window and the Timeline Track Control Panel.

Real-time AAX processing in Media Composer is analogous to the track inserts feature in Pro Tools. For more information about track inserts in Pro Tools, see the [Real-time processing](#) section in the [Pro Tools Guide](#).

12.41.3.2.1 Creating and accessing real-time plug-in instances To insert a plug-in effect on a track in Media Composer, select the track where you want to apply the effect, which insert location you want to use on the track, and the specific effect you want to add to your sequence.

You can also insert a plug-in track effect by dragging an Audio Track Effect template from a bin to your sequence.

To insert an Audio Track Effect plug-in from the Timeline Right-click the Record Track button or the Track Control panel for the track where you want to apply the insert and select AAX Effects *[track number]* > Insert *[a-e]* > *[insert]*.

To insert an Audio Track Effect plug-in using the insert button

1. Click an Audio Effect insert button in the Track Control panel for the track where you want to apply the insert. This opens the Audio Track Effect tool.
2. Click the Select Effect button, and select an Audio Track Effect plug-in effect. Figure 1: Select an insert in the Audio Track Effect Tool

To insert an plug-in using the Effect Palette

1. In the Project window, click the Effects tab. This opens the Effect Palette. Figure 2: The Effect Palette
2. Click the Audio tab.
3. Click an effect category, select the effect you want, and drag it to the segment or to the Audio Track Effect insert button where you want to apply the insert. This opens the Select Insert dialog box. Figure 3: The Select Insert dialog box

Note

You can only insert mono effects on a mono track, stereo effects on a stereo track, and surround sound effects on a surround sound track.

4. Do one of the following:
 - If you want to add a new insert, click an [Empty] insert button.
 - If you want to replace an existing insert, click the appropriate insert button.

The plug-in effect is inserted in the track to which you dragged the effect icon.

To edit an existing Audio Track Effect Plug-In After you insert an Audio Track Effect plug-in on an audio track, you can access the plug-in controls by using the Track Control panel or the Audio Track Effect tool.

Figure 4: Audio Track Effect plug-in inserts in the Track Control panel Figure 5: Audio Track Effect tool: Select Track, Select Insert, and Select Effect buttons (left), Bypass button (center), and Save Effect button (right)

When you select an insert button in the Track Control panel or an effect in the Audio Track Effect tool, the controls for the plug-in appear in the Audio Track Effect tool window.

Figure 6: The Compressor/Limiter Dyn 3 plug-in window displayed in the Audio Track Effect tool dialog box

You can also open the tool by selecting Tools > Audio Track Effect Tool or right-clicking the Record Track button for the track where you want to edit an insert and selecting Audio Track Effect tool. You can use the buttons in the tool to select a specific insert to edit.

To save changes to a plug-in's settings, do one of the following:

- Click the Save Effect icon in the Audio Track Effect tool
- Close the Audio Track Effect tool

12.41.3.2.2 Using Audio Track Effect Templates If you apply an Audio Track effect and make a set of adjustments to it, you can quickly recreate the same sound on other tracks in your sequence or project. You can save an Audio Track effect with its parameter settings to a bin as an effect template. You can then apply the template to other audio tracks at any time.

You can apply an Audio Track effect template with all its parameters directly to an Audio Track Effect insert button in the Track Selection panel or to clips in the Timeline.

To save an Audio Track Effect as a template Do one of the following:

- Click the Save Effect button in the Audio Track Effect tool and drag it to a bin
- Click an Audio Track Effect button and drag it to a bin

A new track effect template appears in the bin, containing the parameter setting information for the effect. The new effect template is identified in the bin by an effect icon. By default, your Avid editing application names the template by the plug-in name.

To apply an Audio Track Effect template to an audio track Do one of the following:

- Drag the Audio Track Effect template from the bin to an insert button in the Track Selection panel
- Drag the Audio Track Effect template from the bin to a segment on the track where you want to apply the effect. The Select Insert dialog box opens so you can select the insert where you want to apply the effect.

This applies the effect to the track.

12.41.4 Compatibility requirements

Media Composer supports 64-bit AAX Native plug-ins beginning in Media Composer 8.1. There are no Media Composer versions that support 32-bit AAX plug-ins, and Media Composer does not currently support AAX DSP plug-ins.

In addition to implementing the client-side AAX API for a supported platform, Media Composer AAX plug-ins must:

1. Be installed to the AAX plug-ins directory
2. Use a valid file name

12.41.4.1 Install directories

AAX plug-ins must be installed in the system's AAX Plug-Ins directory. See [Building your plug-in installer](#) for more information about creating a plug-in installer.

Host Compatibility Notes Some early versions of Media Composer 8 do not search the system plug-ins directory recursively. If your plug-ins are installed into a sub-directory beneath this main directory then they will not be loaded by the affected versions of Media Composer.

Plug-ins that are uninstalled but still present on the system are placed into the "Plug-Ins (Unused)" directory, which is located next to the Plug-Ins directory.

Media Composer will also search for a Plug-Ins directory next to the actual Media Composer application, and this directory will be used if present. This debug feature can be useful for testing specific plug-ins.

12.41.4.2 Plug-in name and file structure

In order to be recognized by AAE, all AAX plug-in bundles must use the ".aaxplugin" file name suffix. On macOS, the plug-in bundle must use this suffix while the binary itself does not require a suffix. On Windows, the plug-in binary (DLL) must use this suffix.

The directory structure of an AAX plug-in bundle is also important. See [.aaxplugin Directory Structure](#) in the [AAX Format Specification](#) document for more information.

12.41.5 AAX feature support in Media Composer

Media Composer supports many of the same AAX features as Pro Tools. However, some features are not available in Media Composer, and other features are managed differently between the two applications. This section describes how Media Composer handles various optional AAX features.

12.41.5.1 Processing configurations

Sample rates Media Composer operates at sample rates of 32000, 44100, 48000, 88200, 96000 Hz, as well as each rate's film pulldown version scaled by a ratio of 1000/1001: approximately 31968, 40959, 47952, 88111, 95904 Hz.

Note

The AAX API does not currently provide a selector for 32 kHz sample rate support

Track formats Media Composer supports only four track formats:

- Mono
- Stereo (interleaved L/R)
- 5.1 surround in Pro Tools order (L, C, R, Ls, Rs, Lfe)
- 7.1 surround in Pro Tools order (L, C, R, Lss, Rss, Lsr, Lsr, Lfe)

Effects will only see these track formats on input.

Note

Plug-ins that support width-changing configurations between supported and unsupported track formats are not compatible with Media Composer

Channel ordering for plug-ins in Media Composer is identical to the channel ordering in Pro Tools. The channel ordering presentation to users may vary from the channel ordering that is used when sending audio buffers to Pro Tools; Media Composer re-orders channels to Pro Tools order prior to presenting the audio to the effect.

12.41.5.2 Preset management

Media Composer stores plug-in presets in several locations within the app. Presets may be stored and accessed through the following workflows:

- Presets can be stored in Media Composer bins by dragging the pink effect icon from the top of the effect editor window into a bin window.
- Track effect presets are stored with their tracks in the sequence
- AudioSuite presets are stored with their audio clips in the sequence

The storage of AudioSuite presets with clips in Media Composer is very different from Pro Tools. To ensure compatibility with Media Composer, it is very important that any AudioSuite effect can be re-rendered from the source media at any time.

12.41.5.2.1 Plug-in preset compatibility and persistence It is always important to design AAX plug-in preset data in a way that will be compatible across different systems and at different points in time. This is particularly true when designing an AAX plug-in to be compatible with Media Composer.

Media Composer sequences carrying presets can be exported as AAF, and these sequences may be moved freely between Media Composer systems on different operating systems and platforms. Therefore, it is important that plug-in preset data is not platform specific. A plug-in loaded in any given Media Composer system must be able to successfully read, parse, and apply preset information that was created on a different system.

Presets also persist for a long time in sequences, so preset information should be formatted in a way that newer plug-ins can read older version's data, and older versions can read newer version's data.

In addition, Media Composer 8.4 and higher can access factory presets and user-created presets interoperably with Pro Tools. A user can save a preset in one application, and access it in the other.

These preset compatibility considerations also apply to plug-ins carried over from legacy plug-in formats such as TDM/RTAS. Media Composer 8.1 and higher (with 64-bit AAX support) will match plug-in IDs when loading sequence data saved with Media Composer 7 and below, which use older plug-in formats. The same system is used for matching plug-in IDs when moving presets between different versions of Pro Tools, and between Pro Tools and Media Composer: in all cases, a preset saved for a particular plug-in ID must be compatible with all other plug-ins that use that ID, regardless of the plug-in format.

12.41.5.2.2 Plug-in preset data comparison Media Composer's rendered AudioSuite effect feature relies on a comparison of plug-in settings chunk data. Unlike in Pro Tools, this operation uses direct data comparison rather than `AAX_IEffectParameters::CompareActiveChunk()`. Therefore, Media Composer compatibility and proper operation of AudioSuite rendering in Media Composer depends on the plug-in having fully consistent AAX preset contents from one run to the next.

Two specific areas where problems can occur are:

- Uninitialized memory in the preset chunk data
- Floating point values in the preset chunk data

Both of these can result in differences between settings chunks representing the same plug-in state, which causes Media Composer to perpetually re-render the plug-in.

The problem of uninitialized memory is obvious. Given a particular plug-in state, Media Composer expects that any retrieved settings chunk will contain matching data regardless of when the chunk is retrieved. When the chunk contains uninitialized data this data does not match between different retrieved chunks. The fix, of course, is to make sure the entire chunk is initialized, for example by setting the entire chunk to zeroes before filling in the data.

The problem of floating point values is more subtle. Depending on the plug-in's parameter implementation, floating point values may be slightly different in the lowest-order bits when set onto the plug-in as part of an incoming chunk and when subsequently read out. When this occurs, Media Composer sees a mismatch in the chunk data, which causes the AudioSuite plug-in to unexpectedly be seen as requiring a new render.

AAX plug-in developers will need to avoid both of these conditions in order to maintain compatibility with Media Composer's AudioSuite effect rendering model.

12.41.5.3 Unsupported features

The following AAX features are not supported by Media Composer. Plug-ins that require these features will not be compatible with Media Composer. If your plug-ins use these features for advanced functionality but not for basic operation then you should document this restriction for Media Composer users.

- Advanced audio routing Media Composer has a simplified audio topology with only tracks and a single master fader. There are no side chains, no busses, and no submasters. As a result, Media Composer does not support extended routing options such as [Sidechain Inputs](#) or [Auxiliary Output Stems](#)
- Transport interface Media Composer does not fully support the [AAX_ITransport](#) interface. In addition, early versions of Media Composer 8 that do not support this interface at all may incorrectly return [AAX_SUCCESS](#) to method calls on this interface. More recent versions of Media Composer will either provide valid information or return [AAX_ERROR_UNIMPLEMENTED](#).
- MIDI Media Composer does not support MIDI routing to and from plug-in instances, and no [AAX MIDI features](#) are supported by Media Composer.
- External control surfaces Although Media Composer does support external control surfaces for some editing functions, it is not currently possible to control plug-in parameters using external control surface hardware in Media Composer.

12.41.5.4 Additional feature support notes

- [AAX](#) plug-in notification support varies between Media Composer and Pro Tools. Media Composer does not support the following notifications, and may not support additional notifications as well:
 - [AAX_eNotificationEvent_ASProcessingState](#)
 - [AAX_eNotificationEvent_ASPreviewState](#).
 - [AAX_eNotificationEvent_SessionBeingOpened](#)
- Media Composer does not support AudioSuite rendering to a separate track (see [AAX_eProperty_DestinationTrack](#))

12.41.6 Additional Information

12.41.6.1 Audio Engine features and behavior

Media Composer shares the same audio engine as Pro Tools (AAE) and both applications share the same advanced audio processing features. However, some aspects of plug-in operation are different between the two apps.

Here are some important notes regarding how Media Composer handles plug-in instances within the audio engine:

- Media Composer only runs plug-ins when Media Composer is playing. Unlike Pro Tools, Plug-ins stop processing when Media Composer stops playing.
- Media Composer buffer sizes are always 1024 samples, and execution is not strictly linked to real-time. Processing is generally between four and eight frames ahead of when the audio is heard.
- Media Composer will render, mix down, and export real-time effects as fast as the processor will allow, typically much faster than real-time, so be careful of introducing real-time dependencies.
- Media Composer has a background render capability, so you cannot expect the GUI to be available, or even be possible on the system performing the render.
- Plug-ins are frequently disposed and re-created on their preset data. This happens with every edit that changes the number, length, or position of playable clips in the timeline.

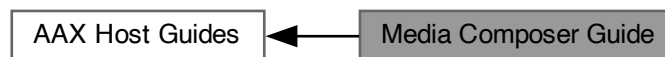
For more detailed information about how AAE handles plug-in loading and processing, see [Audio Engine Behavior and Features](#) in the [Pro Tools Guide](#).

12.41.6.2 Debugging AAX plug-ins in Media Composer

Media Composer does not support attaching a debugger in order to debug plug-ins while they are loaded within the app. In addition, Avid does not currently provide debuggable "developer build" versions of Media Composer. You must therefore rely on logging information for debugging your plug-ins in Media Composer, or debug your plug-ins using other AAX hosts such as a Pro Tools development build or the DigiShell command-line environment.

For more information about debugging in Pro Tools and DigiShell, see [Debugging AAX plug-ins](#) in the [Pro Tools Guide](#).

Collaboration diagram for Media Composer Guide:



12.42 HDX DSP Guide

How to write AAX plug-ins for Avid's TI DSP-based platforms.

12.42.1 Contents

- [Overview of TI DSP Algorithms in AAX](#)
- [Getting Started with HDX DSP](#)
- [The HDX DSP Platform](#)
- [Requirements for HDX DSP Plug-Ins](#)
- [TI Development Tools](#)
- [Common Issues with TI Development](#)
- [TI Optimization Guide](#)
- [Error Codes](#)

12.42.2 Overview of TI DSP Algorithms in AAX

Avid's hardware-accelerated audio systems allow AAX plug-ins to offload their real-time processing tasks to a dedicated processor, guaranteeing reliable performance at ultra-low latency. Avid's TI DSP-based products utilize Texas Instruments DSP chips to host plug-ins in a managed shell environment.

The AAX host handles all system-level communications and resources on the DSP and provides a consistent API to manage communication between the plug-in's real-time algorithm and its other components. This design allows AAX plug-ins to use the same communication methods whether they are running natively, on a TI-based accelerated system, or in some other distributed environment.

Each AAX plug-in contains a real-time algorithm callback. For TI DSP-based platforms, this callback is compiled into a relocatable ELF DLL. This library is loaded onto the appropriate DSP by the host, and may share the DSP with other plug-ins if the host determines that the required system resources are available. A real-time execution environment called the TI Shell is also loaded onto each DSP. The TI Shell manages the DSP's memory and interrupts and guarantees reliable real-time performance even at single sample operation.

12.42.3 Getting Started with HDX DSP

This section provides a quick overview of what you will need for creating an AAX DSP plug-in to run on Avid's TI-based HDX DSP platforms.

- Plug-in structure

The algorithm component for an AAX DSP plug-in is compiled into a binary that runs on the TI DSP. Because this algorithm callback runs on a separate device than the rest of the plug-in, the algorithm must be separated from the plug-in's other components and no pointers may be shared between the two. All memory used by the algorithm must be set up via fields in the algorithm's context structure, and the AAX packet system must be used for transmitting coefficients from the plug-in data model to the algorithm.

For more information about the structure of an AAX plug-in algorithm and features for communicating with the algorithm, see [Real-time algorithm callback](#).

- Development Environment

To compile your plug-in's AAX DSP binary you will need to run TI's free Code Composer Studio (CCS) IDE on a Windows system or VM.

The latest Code Composer Studio versions no longer come bundled with compilation tools compatible with C6727 DSPs, so you must use download and install version 7.x from <https://www.ti.com/tool/download/C6000-CGT/7.4.24>.

To get Code Composer Studio version 12, visit <https://www.ti.com/tool/ccstudio> and navigate to Downloads > View all versions

For additional steps to set up Code Composer Studio see [TI Development Tools](#)

- Language support

The C6727 compiler for AAX DSP plug-ins supports C and C++ up to C++98. In particular, note that no C++11 or later language support is available. For more specific details see [C++ standard support](#).

12.42.4 The HDX DSP Platform

HDX DSP is Avid's core mixer and plug-in accelerator platform. Avid's HDX and Pro Tools | Carbon systems both use the HDX DSP platform, with multiple TI C6727 DSPs each clocked at 350 MHz. These DSPs utilize a 32-bit floating-point architecture, with the option to perform 64-bit double-precision operations at some performance cost. Each HDX card includes 18 DSPs and is connected to the host system over a high bandwidth PCIe connection, while each Pro Tools | Carbon system includes 8 DSPs and is connected to the host system over a Gigabit Ethernet connection.

12.42.4.1 DSP characteristics: instruction processing

The C6727 DSP utilizes a VLIW architecture and contains dual data paths. Each data path includes four independent functional units, so the DSP can accommodate up to 8 parallel instructions per cycle. To take advantage of this architecture, the TI compiler relies heavily on instruction pipelining for optimization.

12.42.4.2 DSP characteristics: audio buffers

In order to realize the maximum possible performance benefit from this architecture, the algorithm routine for a single HDX DSP plug-in is always called with the same buffer size. By guaranteeing that each algorithm will be called with a consistent buffer size, the TI compiler is able to properly account for any possible iterative instruction pipelining, resulting in large performance gains.

HDX DSP uses a four-sample processing quantum by default for plug-in instances. Plug-ins that require additional processing time per callback, e.g. to mitigate the overhead cost of the chip's DMA facilities, may optionally request a 16, 32, or 64-sample quantum. Note that at higher block sizes, the number of potential I/O channels available to plug-ins on a chip will be reduced.

Host Compatibility Notes 32 and 64-sample quantum is available in Pro Tools 10.2 and higher

12.42.4.3 DSP characteristics: memory

Each DSP on the HDX DSP platform includes 16 MB of external RAM and 256 kB of internal RAM. The DSP has the ability to execute code from either internal or external RAM, though the real-time performance cost of external RAM accesses is significant. The chip's internal RAM is addressable at the core clock rate.

Each DSP also has a program cache of 32 kB. Plug-in code is loaded into this cache from internal memory, so for best performance your plug-in should not use more than 32 kB for its program code. You can look at the CCS-generated .map file to find your plug-in's program code size.

12.42.4.3.1 SDRAM performance Asynchronous access to data in the C6727's SDRAM is very slow, requiring 50 cycles/word to read and 15 cycles/word to write. This is primarily due to clock domain bridging, lack of data caching, and the fact that data from the core is given a low priority in order to avoid stalling real-time DMA transfers.

12.42.4.3.2 Executing program code from external memory The TI C6727 supports executing program code from external memory. When executing from uncached external memory, expect cycle counts to increase by a factor of 4x to 5x compared with the equivalent internal-memory code. Assuming that no cache thrashing occurs, subsequent calls will be cached and thus the program's location in either external or internal memory will produce similar cycle counts.

Note

The CCSv4 Profiler contains a bug that produces incorrect cycle counts for cached external-memory program code. Therefore, when gathering cycle count data for a plug-in that stores its program data in external memory, an RTI-based timing method should be used.

12.42.4.4 System characteristics: DSP/host data transfers

Plug-ins loaded onto the HDX DSP platform may transfer arbitrarily large data blocks between the DSP and the host, within the limits of available DSP memory and system bandwidth.

12.42.4.4.1 DSP/host bandwidth Neither AAX nor the HDX DSP platform include any explicit plug-in bandwidth limiting constraints. If a plug-in's data transfer requests bump up against the physical bandwidth limit for the system then this will delay the blocking data transfer request on the host, as the transfer will be held off for higher-priority operations on the DSP, and may also delay automation data from reaching other plug-ins on the affected DSPs in the same group.

The recommended upper limit for DSP/host data transfer requests in an individual plug-in when running on an HDX PCIe card is 10 MB/s, divided by the maximum number of plug-in instances that will run on a single chip. On the HDX card, DSPs are wired to the FPGA crossbar in groups of three, with a data bandwidth of approximately 67 MB/s for each group. The overall system bandwidth for each DSP is therefore approximately 20 MB/s. This bandwidth is shared by all data reads and writes, including custom data transfer requests as well as plug-in and mixer automation and metering data.

This limit is significantly lower on Pro Tools | Carbon. Carbon uses a single group for all eight DSPs, so the overall system bandwidth for each DSP is approximately 8 MB/s. In addition, data transfers between Carbon and the host system must be executed over a Gigabit Ethernet connection with up to 75% of its bandwidth already reserved for AVB audio data. This leaves 250 Mb/s for all other command traffic. If your plug-in utilizes frequent or large DSP/host data transfers then be sure to test it on Pro Tools | Carbon to verify whether it is compatible.

12.42.4.4.2 DSP/host data transfer characteristics The minimum data transfer size for all host-to-DSP communications for HDX PCIe cards is 128 bytes. This limit applies to all host-to-DSP data transfers, including data sent to buffered ports, unbuffered ports, and private data blocks (via the AAX Direct Data interface.)

Since each transfer has a minimum size of 128 bytes, the use of many small packets does not increase transfer efficiency or save system bandwidth. Quite the opposite: updating a single 64-byte packet would require less bandwidth than updating two 4-byte packets in an HDX PCIe system, since the former would require only one 128-byte transfer while the latter will require two.

On Pro Tools | Carbon there is no minimum data transfer size. That said, for best performance on Carbon it is still recommended to minimize the number of data packets that are sent.

12.42.4.5 TI Shell characteristics: Memory allocation

12.42.4.5.1 Memory resource availability The TI Shell code that is loaded onto each DSP uses approximately 56 kB of internal memory, leaving 200 kB of internal memory per DSP. This memory is shared between the plug-ins on the chip and holds the plug-ins' code and data, per-instance blocks declared in `Describe()`, and instance overhead.

As a general guideline, plug-in instances should not use more than $200 / n$ kB of internal memory, where n is the number of instances of your plug-in that will run on a single chip based on its cycle count requirements. If each plug-in instance on the chip requires more internal memory than this then the plug-in may need to declare an explicit number of instances that can run per chip based on this memory usage rather than declaring its cycle count utilization.

12.42.4.5.2 Shared and per-instance memory allocation When a plug-in instance is created on a DSP, its program code is loaded onto that DSP. This copy of the program code is then re-used for all subsequent instances of the effect that are loaded onto the DSP. Static and global data are also shared between all instances of an effect on the DSP. Other allocations, such as coefficient and private data blocks, are per-instance.

Host Compatibility Notes Beginning in Pro Tools 11, AAX DSP algorithms also support optional temporary data spaces that can be described in the `Describe` module and are shared among all instances on a DSP. This is an alternative to declaring large data blocks on the stack for better memory management and to prevent stack overflows. Please refer to [AAX_IComponentDescriptor::AddTemporaryData\(\)](#) for usage instructions.

12.42.4.5.3 Placing data into external memory An AAX plug-in may optionally request that its private data or program code be placed into external memory. Because standard access calls to the DSP's SDRAM are very slow, it is strongly recommended that all of a plug-in's real-time data be placed in internal RAM, and the TI Shell will load a plug-in's program code and all private plug-in data blocks into internal memory by default.

Requesting more than 256 kB of data in internal memory for plug-in data plus the memory required by the TI Shell will lead to undefined behavior, so it is important to explicitly request external memory for plug-in data when appropriate.

For private data blocks that should be loaded into external memory, use the [AAX_ePrivateDataOptions_External](#) flag when calling [AAX_IComponentDescriptor::AddPrivateData\(\)](#). This flag will be ignored by the host, so Native AAX plug-ins will have the same functionality with or without this property.

To load program code, static data, or global variables into external memory, use the `TI_SECTION` pragmas. For example, `#pragma CODE_SECTION_(".extmem")` can be used before function definitions that are either initialization code, or infrequently used background code. For static variables, use `#pragma DATA_SECTION_(".extmemdata")` before each variable definition.

12.42.4.5.4 DMA support Because of the slower access time of external RAM, you should consider using a [DMA transfer](#) for recurring transfers, and possibly even for larger one-time transfers. This is of particular relevance for data reads, which must traverse the various clock domains and priority switches twice (address send, and then data return.)

The TI Shell supports three DMA modes: Scatter (for transfers from internal to external memory), Gather (for transfers from external to internal memory), and Burst (contiguous block copies). The Scatter mode can accomplish transfer speeds of up to 2.1 DSP cycles/byte transferred, while the Gather mode can accomplish 2.7 cycles/byte transferred.

The Scatter and Gather DMA facilities use a linear buffer for internal memory and a FIFO for external memory. It is possible to transfer to or from multiple offsets within the external memory FIFO using an offset table, which can contain up to 65,536 (2^{16}) entries. The offset (burst) length may be 4, 8, 16, 32, or 64 bytes long.

The TI Shell also supports a Burst DMA mode which implements linear data reads or writes.

For more information on DMA support and for example code, see `\ExamplePlugIns\DemoGain_DMA` in the SDK.

12.42.4.6 TI Shell characteristics: Data packet services

In addition to supporting direct transfers of arbitrary data via DMA, the TI Shell also supports a packetized data delivery mechanism for host-to-DSP data transfers. Packet delivery ports may be either unbuffered or buffered, and are described using the `AAX_EDataInPortType` parameter in `AAX_VComponentDescriptor::AddDataInPort()`.

12.42.4.6.1 Unbuffered ports Unbuffered ports use a straightforward implementation that delivers posted packets to the algorithm as soon as possible. In an unbuffered port, newer packets will always override older packets. Therefore, an algorithm may not receive every packet that was posted to an unbuffered port, but it will always receive the most up-to-date information possible.

Unbuffered ports deliver their data without blocking or synchronizing with the algorithm's execution. Although bus arbitration guarantees that a read from the algorithm callback will not occur in the middle of a write from the host, it is important to note that the data in an unbuffered port may change during algorithm execution.

12.42.4.6.2 Buffered ports Buffered data ports store incoming packets in a host-managed queue. This queue acts as a buffer and provides the host with more flexibility in how it delivers packets. A key feature of buffered data ports is that new data will never be delivered to these ports during algorithm execution.

The behavior of buffered data ports varies depending on the host platform. In HDX DSP plug-ins, Buffered data ports use a FIFO to queue data packets as they are posted. New packets are dequeued and delivered to the algorithm individually, with the next packet arriving before each algorithm render callback.

12.42.4.6.3 Data port overhead and restrictions Each HDX DSP supports a maximum of 164 buffered data ports, which matches the maximum I/O limit for each DSP. System overhead costs associated with using the on-chip packet services are as follows:

Memory Overhead

- The memory overhead for an unbuffered data port is simply the size of the data packet.
- This DSP memory overhead for a buffered data port is two times the size of the data packet. A large (>100-element) packet queue is also allocated on the host.

CPU overhead Unbuffered ports do not incur any additional CPU overhead.

Individual buffered ports incur non-trivial CPU overhead. For example, in Pro Tools 10.2 each buffered port requires 5 cycles of overhead per render callback. This overhead can quickly add up in "small" plug-ins that contain many buffered data ports. Therefore, we strongly recommend that plug-ins use consolidated coefficient packets when possible in order to minimize this overhead. This optimization can result in large performance gains for callbacks that require 1000 or fewer cycles to operate.

The trade-off of this optimization is that more work ends up being done on the host and more data must be transmitted to the algorithm, since the entire coefficient packet must be re-calculated and re-sent every time any of its input parameters change. This is usually beneficial trade-off to make, especially given the 128-Byte per-transfer minimum for HDX PCIe cards discussed above. However, care must be taken in extreme cases such as when packet delivery threatens to bump up against the maximum recommended bandwidth for host/DSP data transfers, especially on Pro Tools | Carbon.

12.42.4.7 TI Shell characteristics: Instance allocation

12.42.4.7.1 Multi-shell packing With a few exceptions, AAX DSP plug-ins will share DSPs with other plug-ins. This occurs transparently to the plug-in due to the fact that all system resource management is handled by the TI Shell.

When a new plug-in instance is created, the TI Shell and AAX host will attempt to intelligently allocate it to a DSP based on both memory and CPU resource requirements. If one plug-in on the chip requires a large amount of memory and very few processing cycles, it may be packed with another plug-in that does not require much memory but that is very CPU intensive.

Each DSP chip runs audio callbacks at a single buffer size, so plug-ins that run at different buffer sizes will not be loaded onto the same DSP chip. For example, if a plug-in processes at 16 sample buffers then it will only share a chip with other plug-ins that process at 16 sample buffers. An AAX DSP plug-in's buffer size is defined by its [AAX_eProperty_DSP_AudioBufferLength](#) property. Most plug-ins use the [default](#) of 4 sample buffers, and that is the value that will maximize chip sharing.

Certain plug-ins cannot share a DSP with other plug-ins:

- Plug-ins that use [DMA](#)
- Plug-ins that register for a [background processing](#) callback
- Plug-ins that register a maximum number of instances per chip using [AAX_eProperty_TI_MaxInstancesPerChip](#)

These plug-ins will receive dedicated DSPs to which only additional instances of the same plug-in type will be added.

The TI shell also supports a [processor affinity](#) property, which indicates that a DSP ProcessProc should be preferentially loaded onto the same DSP as other instances from the same DLL binary. This is a requirement for some designs that must share global data between different processing configurations.

12.42.4.7.2 DSP Shuffles A DSP shuffle will occur in Pro Tools when the engine must re-allocate DSP resources in order to make more processing power available. A shuffle will force the re-instantiation of the plug-in's DSP algorithm component, potentially on a new chip, while leaving the plug-in's host objects intact. During a shuffle, the engine will perform the following steps:

1. Disconnect audio from an effect
2. Call instance initialization with the removing instance flag on the old location
3. Repeat for all instances of all DSP Effects in the system
4. Load the effect in the new location
5. Re-send the last packets to all data-in ports
6. Call private data init for any private data
7. Call instance init with the 'adding instance' flag, in the new location
8. Begin audio processing
9. Reconnect audio
10. Repeat the instantiation and connection process for all instances of all DSP Effects in the system

Note that the system may perform some audio processing with each new instance before all of the Effect instances in the system have been re-instantiated.

12.42.4.8 Additional TI Shell services

12.42.4.8.1 Background processing AAX plug-ins may request idle time from the main TI Shell thread. This results in a true idle context callback which can be used for non-critical [background processing](#) tasks on the DSP. This facility restricts the DSP to only allocate plug-in instances of the same type.

A plug-in's background processing callback is not provided with a reference to the plug-in's data structures and must therefore access plug-in data via global variables. The background process will be interrupted by system events and the audio render callback. For more information and an example on how to create a plug-in that relies on background processing, see `\ExamplePlugins\DemoGain_Background` in the SDK.

12.42.5 Requirements for HDX DSP Plug-Ins

12.42.5.1 Plug-in description

To support HDX DSP platforms, a plug-in must add a TI ProcessProc (real-time processing entrypoint) for each of its algorithms. This is done via a call to [AAX_IComponentDescriptor::AddProcessProc_TI\(\)](#), which is parametrized with the names of both the algorithm's TI DLL and of its exported entrypoint.

At minimum, the TI ProcessProc requires the following AAX Properties:

- A TI plug-in ID: [AAX_eProperty_PluginID_TI](#)
- The audio buffer size that will be used by the ProcessProc: [AAX_eProperty_DSP_AudioBufferLength](#), set with a value from [AAX_EAudioBufferLengthDSP](#)

12.42.5.2 Performance measurement and reporting

In order to determine each algorithm's resource requirements, the host collects cycle count information from the plug-in via the plug-in's Describe callback. Each plug-in Effect is responsible for correctly reporting its algorithms' cycle counts for each accelerated platform that it supports. For plug-ins that use DMA or background threads, a maximum per-chip instance count is also required.

Note

All reported values must represent the algorithm's worst case performance.

Each of these values are reported as properties of a given algorithm ProcessProc and are provided by the plug-in via [AAX_IComponentDescriptor::AddProcessProc_Tl\(\)](#). If an effect does not report its cycle count usage then it will be limited to a single instance per TI chip. This can be useful during development, but is not a supported mode for general use; all shipped plug-ins must correctly report their cycle requirements.

The DigiShell utility can be used to accurately measure plug-in cycle count requirements. For more information about DigiShell, see [DSH Guide](#).

12.42.5.2.1 Shared vs. per-instance cycles Because a single call into a plug-in is used to process multiple instances of that effect on that chip, two cycle count properties must be reported for each TI algorithm:

1. [AAX_eProperty_Tl_SharedCycleCount](#) This property describes the algorithm's one-time processing overhead that doesn't change as instances are added to a chip.
2. [AAX_eProperty_Tl_InstanceCycleCount](#) This property describes the additional cycle counts that each instance adds to the base shared overhead.

Many plug-ins exhibit different performance characteristics for both of these metrics depending on the plug-in's state. When reporting a plug-in's shared and per-instance cycle count requirements it is important to ensure that the reported values are the *maximum possible requirements* of the algorithm.

Often a plug-in will experience its worst-case per-instance processing load in one configuration and its worst-case shared processing load in another configuration. In this situation, the plug-in's reported cycle count requirements should reflect the state in which the *sum* of the two metrics is highest.

It's a common practice to not describe [AAX_eProperty_Tl_InstanceCycleCount](#) and [AAX_eProperty_Tl_SharedCycleCount](#) for the plug-ins during development and debugging process of the DSP plug-ins. This is acceptable, although in this case the one instance of such a plug-in will require the whole chip. In AAX SDK example plug-ins this is implemented using `AAX_TI_BINARY_IN_DEVELOPMENT` macros. If defined, it turns off the cycle count properties for the plug-in.

12.42.5.2.2 Measuring shared cycles Measuring shared cycle counts requires instantiating multiple instances of an effect and observing how the processing time changes as instances are added. The shared and instance cycle counts are then calculated by performing a linear regression on the number of uncached cycle counts as the number of plug-in instances on the chip increases.

Note that these values will differ between debug and release builds of an algorithm, so a plug-in's describe function should report the correct cycle count values based on the relevant build configuration.

DigiShell includes the ability to measure shared cycle counts using the `DAE.cyclesshared` command. For more information about performance profiling using DigiShell, see [Cycle count performance test](#).

Note

HDX DSP requires reporting of an algorithm's *worst-case* cycle counts.

Because HDX PCIe and Pro Tools | Carbon use the same HDX DSP platform, either product may be used to take plug-in cycle count measurements.

12.42.5.2.3 DMA and background thread performance reporting For algorithms that use [DMA](#) or [background thread](#) facilities, the maximum number of algorithm instances that will fit on a chip is difficult to predict from cycle counts alone. Due to the asynchronous behavior and limited capacity of the DMA system, the DMA system may begin to miss its deadlines before the CPU is fully loaded. In addition, due to differences in background processing requirements between algorithms, an effect's background process may begin to miss its deadlines and be starved before the interrupt-time audio processing is at capacity. Plug-ins that use these facilities must therefore report the maximum number of instances that will run reliably at a given sample rate, in addition to reporting their shared and per-instance cycle counts as above.

Some other plug-ins may also wish to report the maximum number of instances that will run reliably at each sample rate. For example, plug-ins that use a lot of host/DSP data bandwidth may need to limit the number of instances per DSP chip in order to run successfully on Pro Tools | Carbon.

Maximum reliable instance counts are reported using an additional property, [AAX_eProperty_TI_MaxInstancesPerChip](#). A plug-in should register separate components for the following three sample rate ranges in order to register distinct values for this property:

1. Sample rates from 42kHz to 50kHz
2. Sample rates from 84kHz to 100kHz
3. Sample rates from 168kHz to 200kHz

Notes regarding DMA and background thread performance reporting:

- Because the number of instances will decrease as sample rate increases, the plug-in must be tested at the highest available pulled-up sample rate (i.e. 50kHz instead of 48kHz) in each of these three ranges.
- On the HDX platform, effects that use DMA or background threads will not be mixed with effects of other types on a given chip.
- The maximum number of instances per DSP cannot be measured via DSH in these cases, so careful listening tests must be manually performed in order to determine whether a certain number of instances of a DMA or background-enabled plug-in actually operate correctly on a DSP.

12.42.5.2.4 Dynamic resource usage All resources used by an AAX DSP plug-in algorithm are considered static. Plug-ins may not dynamically change the amount of memory or DSP cycles that are allocated to them after these metrics are provided in `Describe`.

The ability to dynamically change DSP cycle count requirements at run time is provided in the AAX SDK but is not currently supported by any host.

12.42.5.3 Plug-in compilation and packaging

12.42.5.3.1 Exported symbols Each HDX DSP algorithm (ELF DLL) may contain multiple entrypoints. A single DLL may be used for all of your plug-in's entrypoints and program code, or you may divide your plug-in's entrypoints and program code between multiple DLLs.

Your plug-in must export one "C"-style callback for each algorithm `ProcessProc` that your plug-in registers. This entrypoint must conform to the standard AAX real-time algorithm callback prototype:

```
# include "elf_linkage_aax_ccsv5.h" // Includes required TI_EXPORT definition
extern "C"
TI_EXPORT
void
MyEffect_AlgorithmProcessFunction(
    SMyEffectAlg_Context * const inInstancesBegin [],
    const void * inInstancesEnd)
```

Listing 1.1: The standard AAX real-time algorithm callback prototype

See [Settings for exported symbols](#) if you are running into problems linking your AAX DSP binary.

12.42.5.3.2 Packaging The ELF DLLs for an AAX DSP plug-in must be placed in the ./Content/Resources directory within the plug-in bundle.

12.42.6 TI Development Tools

Development for TI algorithms is primarily performed in TI's Code Composer Studio. Code Composer Studio (CCS) is a full-featured, Eclipse-based IDE providing JTAG hardware debugger support, a hardware simulator, and a suite of profiling tools. Most importantly, CCS includes an excellent C compiler that is capable of providing highly optimized DSP instructions without too much tuning.

Note

As of this writing, Code Composer Studio for Mac does not support the C6000 series processor. CCS for Windows is required for AAX DSP plug-in development. See [MacOS Host Support CCSv7](#) on the Texas Instruments wiki for current compatibility information.

12.42.6.1 Code Composer Studio

The AAX SDK supports Code Composer Studio versions 4 ("CCSv4") and higher ("CCSv5", etc.), with hardware debugging support beginning in version 4.2. As of the writing of this documentation, CCS versions 4, 5, 7 and 12 have been tested by Avid.

Note

This documentation was originally written for CCSv4 and was later updated with instructions for updating from CCSv4 to CCSv5. Versions 5 and higher use a different project file format from version 4; when this documentation describes changes required for version 5 then these changes will also be required by other later versions which use this new project format.

12.42.6.1.1 Installation

1. Download and install the latest Code Composer Studio from TI's website.

Note

Windows 10 requires Code Composer Studio version 6.1.3 or higher

As of Code Composer Studio version 7 TI does not charge for licenses. You can simply download the tool and start using it. Along with this the end user license agreement has changed to a simple TSPA compatible license. For more information see the TI web site.

2. The default installation will work fine, but a custom install will be smaller. You only need support for the C6000 chipset and, if you have an HDX board with JTAG pins, the Spectrum Digital JTAG drivers, so you can deselect all the other chipsets and JTAG drivers.
3. Download and install the C6000 Code Generation Tools v7.x:
 - (a) Launch CCS and go to Help > Install New Software...
 - (b) In the opened dialog select "Code Generation Tools Updates" in the "Work with:" drop-down list.
 - (c) Select "TI Compiler Updates" > "C6000 Compiler Tools [version 7.x]".
 - (d) Press Next and continue installation using the "typical" installation settings.

As of the publishing of this version of the AAX SDK Avid is internally using v7.4.24. Avid has tested 7.4.4, 7.4.6 and 7.4.24, all later v7 revisions should work as well. CGTools versions higher than v7 do not support the C6727 DSP.

12.42.6.1.2 Workspace setup Each time you launch CCS you will be prompted to select a workspace directory. A CCS workspace is similar to a Visual Studio solution file. Note that workspaces tend to store absolute paths and developer-specific info, so you may wish to avoid checking them in to your source control server.

Setting up workspace-global macros *To set up workspace global macros:*

1. When you open CCS for the first time, select a directory for your "workspace". As mentioned above, we recommend that this be outside of your source tree.

Note

If you are updating from a previous CCS version you may not be able to reuse your workspace. For example, we have found the CCSv4 workspaces are incompatible with CCSv5. After updating your system to a later Code Composer Studio version you should create a new workspace and import your existing projects into this new workspace.

2. Go to File > Import... and select Code Composer Studio > Build Variables (CCS > Managed Build Macros in CCSv4.) Click Next.
3. Browse to TI/Common/macros.ini in your AAX SDK directory and click Finish.
4. This will define an "SDK_SOURCE_ROOT" Linked Resource path variable and Managed Build macro, which associates the CCS workspace with a single AAX SDK installation.

Note

A side effect of this is that you cannot use projects from multiple distinct AAX SDK installations in the same CCS workspace.

5. To verify that the correct path has been set, go to Window > Preferences... and look in General > Workspace > Linked Resources, and C/C++ > Build > Build Variables (C/C++ > Managed Build > Macros for CCSv4.)

Importing projects into your workspace *To import projects into your workspace:*

1. In the IDE, go to Project > Import CCS Projects...
2. In Select search-directory, select the root of your AAX SDK installation.
3. The CCS projects in the AAX SDK will be added to the Discovered projects field. Select All or choose the specific projects you want to import.
4. Click Finish, and then wait while the projects are imported.

In order to import CCSv4 projects into later versions of Code Composer Studio it is necessary to add a .cdtproject file to the project. If you don't have this file in your project, then you can copy it from any other existing project which was created using CCSv5 or later. Otherwise you will most likely see something similar to this error:

"Error: Import failed for project 'xxxx' because its meta-data cannot be interpreted."

If you try to build this newly imported CCSv4 project in a later version of Code Composer Studio then you will get the warning:

"This project was created using a version of compiler that is not currently installed: 7.0.5 [C6000]. Another version of the compiler will be used during build: 7.4.24. Please install the compiler of the required version, or migrate the project to one of the available compiler versions by adjusting project properties."

This warning may be cleared by changing Properties > General > Compiler Version from TI v7.0.x to the current version (e.g. TI v7.4.x). After that the "Output format" field, which is next one to the "Compiler version" field and is typically grayed out, will become active. You should choose "eabi (ELF)" there. Otherwise Code Composer the build will fail with errors:

- "--dynamic=lib not supported when producing TI-COFF output files"
- "--export=_auto_init_elf not supported when producing TI-COFF output"

Note

After successful conversion of the project and successful build, the remeasurement of cycle count should be done, because it may change. Most likely it will decrease, as compared to the version which was built with CCSv4, but that is not guaranteed. Also the size of the DLL may increase, which may require reducing code size in order to properly instantiate the plug-in.

12.42.6.1.3 Creating new projects

New project setup Use the following settings in the "New Project..." wizard. Defaults are in *italics*.

- Project Type: C6000
- Output type: Executable
- Device Variant: Generic C67x+ Device
- Device Endianness: little
- Code Generation Tools: 7.4.24 or later (7.0.5 for CCSv4)
- Output format: eabi (ELF) (in CCSv4 this field will be grayed out.)
- Linker Command File: CommonPlugIn_LinkersCmd.cmd (see note below)
- Runtime Support Library: <automatic>

Note

You can edit the Linker Command File setting to use the `SDK_SOURCE_ROOT` macro by manually editing the project's .project XML file or by adding the file to your project using a relative path. See the SDK sample plug-in projects for an example.

12.42.6.1.4 Recommended settings for AAX plug-in projects Tool Settings C6000 Compiler Include Options

```
-include_path "${SDK_SOURCE_ROOT}/Interfaces" -include_path "${SDK_SOURCE_ROOT}/[Plug-in directory]"
```

The `SDK_SOURCE_ROOT` macro is defined via the macros.ini file, located in the SDK's /TI/CCSv4 directory. If you encounter errors using this macro, import the file using File > Import... > CCS > Managed Build Macros.

Tool Settings C6000 Compiler Command Files -cmd_file "\${SDK_SOURCE_ROOT}\\TI\\CCSv4\\CommonPlugIn_CompilerCmd.cmd"

This file contains additional compiler commands that should be common to all AAX plug-in projects

Tool Settings C6000 Linker Basic Options `-o "${ConfigDir}/${PackageName}/Contents/Resources/${ProjName}.dll"`

This path will ensure that your compiled TI DLL is placed in the appropriate location inside your AAX plug-in bundle.

Tool Settings C6000 Linker Runtime Environment (No "Initialization model" options set)

Build Settings Artifact name `${ConfigDir}/${PackageName}/Contents/Resources/${ProjName}`

This path will ensure that your compiled TI DLL is placed in the appropriate location inside your AAX plug-in bundle.

Build Settings Artifact extension `dll`

AAX TI libraries should use the `.dll` extension

Binary Parser Elf Parser

AAX TI libraries should use the Elf binary parser only

Macros Project User Macros `ConfigDir = ${OutDir}/${ConfigName}` `IntDir = ${ConfigDir}/int/${PackageName}/TI/${ProjName}` `OutDir = ${ProjDirPath}/../../WinBuild` `PackageName = [Plug-in name]`

These macros are used by the other settings here to ensure proper path set-up and artifact naming. Don't worry that `ConfigName` shows up as undefined - it will be defined as `Debug/Release` at compilation.

12.42.6.1.5 Recommended Release configuration settings Tool Settings C6000 Compiler Basic Options `-symdebug:none -O3`

Tool Settings C6000 Compiler Predefined Symbols `-define=NDEBUG`

Tool Settings C6000 Compiler Optimizations `-os -on2 -op3`

Tool Settings C6000 Compiler Assembler Options `-keep_asm`

12.42.6.1.6 Other useful project settings Tool Settings C6000 Compiler Predefined Symbols `-define _DEBUG`

This option is useful for differentiating cycle count reporting for Debug vs. Release builds.

Tool Settings C6000 Compiler Directory Specifier `-ft "${IntDir}" -fr "${IntDir}" -fs "${IntDir}"`

Useful for collecting intermediate files

Tool Settings C6000 Linker Basic Options `-m "${IntDir}/${ProjName}.map"`

Useful for placing the map file alongside all other intermediates

Tool Settings C6000 Linker File Search Path `-l (nothing)`

You can exclude `libc.a`, which is included by default, from this option unless you require C library features.

12.42.6.1.7 Adding files and folders In CCS, dragging files into the project, using "Add Files to Project...", or using "Link Files to Project..." will either copy the file into the project directory or create an absolute path to the file. This is usually not the desired behavior. Use the following steps to add a file using a relative path:

1. Right click on the project you'd like to add files to, and select New > File (NOT "Source File" or "Header File").
2. Click "Advanced >>".
3. Check the box that says "Link to the file in the system". Click "Variables..."
4. Select the appropriate variable (usually either `SDK_SOURCE_ROOT` or `SOURCE_ROOT`) and click "Extend..."
5. Find the file you want to add. Click OK. Click Finish.

Note that, when adding folders, *everything* in the folder will be built by default. You can exclude files to work around this behavior.

12.42.6.1.8 Settings for exported symbols

- There is a compiler option in Code Composer Studio that will add an underscore to the exported entrypoint's name. We recommend keeping this option disabled in order to avoid ambiguity between the exported symbol name and the function name as it appears in your source code.
- If you encounter undefined symbol errors when linking to a DSP library that uses a C-style interface then add the extern "C" keyword before the lib function prototypes. This should resolve the majority of such linker errors.

12.42.6.2 The TMS320C6000 C++ compiler

One of the primary goals of AAX is to provide a platform-agnostic development architecture in which products can easily be developed and re-used across a wide variety of platforms. However, it is still occasionally necessary to write platform-specific code. This section will document methods for producing code that is specific to the TI C6727 platform using the TMS320C6000 C++ compiler.

12.42.6.2.1 C++ standard support The TMS320C6000 compiler supports C++ as defined in the ISO/IEC 14882:1998 standard. The exceptions to the standard are as follows:

- Complete C++ standard library support is not included. C subset and basic language support is included.
- These C++ headers for C library facilities are not included:
 - `<clocale>`
 - `<csignal>`
 - `<cwchar>`
 - `<cwctype>`
 - `<ciso646>`
- These C++ headers are the only C++ standard library header files included:
 - `<new>`
 - `<typeinfo>`
- No support for `bad_cast` or `bad_type_id` is included in the `typeinfo` header.

- Run-time type information (RTTI) is disabled by default. RTTI can be enabled with the `-rtti` compiler option.
- The `reinterpret_cast` type does not allow casting a pointer to member of one class to a pointer to member of another class if the classes are unrelated.
- Two-phase name binding in templates, as described in `tesp.res` and `temp.dep` of the standard, is not implemented.
- The `export` keyword for templates is not implemented.
- A typedef of a function type cannot include member function cv-qualifiers.
- A partial specialization of a class member template cannot be added outside of the class definition.

12.42.6.2.2 Predefined environment symbols The following symbols are predefined by the compiler on the TI architecture, and should be used in code concerned with cross-platform support:

- `_TMS320C6X` Identifies that the chip is a C6000 variant. This is the symbol that we commonly use to distinguish whether code is being compiled for AAX-Native (Mac/Windows) or AAX-TI.
- `_TMS320C6700_PLUS` Identifies that the chip is a C6700-plus variant

Although you should not require them for AAX development, equivalent assembly predefines are as follows:

- `.TMS320C6X` Identifies that the chip is a C6000 variant
- `.TMS320C6700_PLUS` Identifies that the chip is a C6700-plus variant

12.42.6.2.3 Loop controls The TI compiler supports several pragmas that can be used to give the compiler additional information about loops.

- `#pragma MUST_ITERATE(min, max, multiple)` This pragma helps the compiler optimize loops. `min` is the minimum number of times the loop will execute, `max` is the maximum number of times the loop will execute, and `modulo` is used if the loop will only execute a certain multiple of some number.
- `#pragma PROB_ITERATE(min , max)` If extreme cases prevent the use of `MUST_ITERATE`, `PROB_ITERATE` allows you to specify the usual number of times a loop executes. For example, `PROB_ITERATE(8)` could be applied to a loop that executes for eight iterations in the majority of cases but that sometimes may execute more or less than eight iterations.
- `pragma UNROLL(n)` Helps the compiler use SIMD instructions, where `n` is the unrolling factor. By specifying `UNROLL(1)` you can prevent the compiler from automatically unrolling a loop. In general, we recommend using `MUST_ITERATE` instead unless you have specifically identified a situation where manually unrolling a loop improves performance.

12.42.6.3 DigiShell test tool (DSH)

DigiShell is a software tool that provides a general framework for running tests on Avid audio hardware. As a command-line application, DigiShell may be driven as part of a standard, automated test suite for maximum test coverage. DSH supports loading all types of AAX plug-ins including Native and DSP, and is especially useful when running performance and cancellation tests of AAX-TI types. DigiShell is included in Pro Tools Development Builds as `dsh.exe` (Windows) or as `dsh` in the `CommandLineTools` directory (Mac).

More information on DSH test tool can be found in [DSH Guide](#).

12.42.6.4 Hardware Debugging

12.42.6.4.1 Requirements Relocatable ELF DLLs (TI algorithms) can be debugged with some help from the DIDL loader, the TI Shell Manager, and a script called `DLLView_Elf_Avid.js`.

These are the minimum requirements for hardware debugging for TI plug-ins:

- Code Composer Studio version 4.2 or later
- XDS510 hardware debugger
- JTAG-enabled HDX card

We recommend using Spectrum Digital's XDS510 USB Plus JTAG Emulator, as it is the only one our internal developers have used and tested in-house. Both Spectrum Digital and TI have useful technical reference/installation guides, both of which can be found on the AAX Developer Forum under the 'Development Tools' discussion.

12.42.6.4.2 How it works The `ridl` ELF loader inside DIDL stores a module and segment list containing the paths of all loaded modules and where their segments are loaded. The TI Shell Manager gets a serialized version of this table and loads it to a block of external memory on the chip at a known location. The `DLLView_Elf_Avid.js` script queries this memory via the debugger and extracts the paths of the modules and the ELF segment load locations, which it then passes on to the `GEL_SymbolAddELFRel` scripting console command (new to CCSv4.2). You can also use that command directly at the console.

12.42.6.4.3 Connecting a JTAG Emulator A JTAG-enabled HDX development card includes a "riser" PCB section extending about a centimeter above the production card PCB. This riser includes two JTAG connectors. The two connectors correspond to the two banks of 9 DSPs on the HDX card. Assuming that you are instantiating your plug-in for debugging on the first available DSP, you will want to connect your JTAG emulator to the connector that is closest to the card's user-visible ports. This connector corresponds to the first 9 DSPs on the card.

12.42.6.4.4 Linking to TIShell.out Hardware debugging, as well as several other debugging facilities, requires that the DSP plug-in project is linked to `TIShell.out` in Code Composer Studio.

To link a plug-in project to TIShell.out, follow these steps:

1. Open the plug-in project's properties window and navigate to the *C/C++ Build > Tool Settings > C6000 Linker > File Search Path* properties pane.
2. Add "TIShell.out" to the "Include library file" (-l) property list.
3. Under "Add <dir> to library search path" (-i), add the file path of the Pro Tools build you will be using to test the plug-in. This directory should already include the build's `TIShell.out` file.
4. Repeat this process for each Configuration of the plug-in project that you will be testing.
5. Add "[path to AAX SDK root]\\TI" to the project's list of source file include directories

12.42.6.4.5 Adding the HDX Target Descriptor File *To add the HDX Target Descriptor File:*

1. In the IDE, go to Window > Preferences, CCS > Debug. Point the "Shared target configuration directory" to `/TI/Common` in your AAX SDK source tree
2. In the IDE, go to Window > Show View > Target Configurations.
3. Click refresh if you don't see the configuration file
4. Right click `Raven_C672x_XDS510_USB.ccxml`, and click "Set as Default".

12.42.6.4.6 Setting up the DLLView script Once you have successfully installed the XDS510, you will have to do a little bit of setup with CCS. Before starting this process, verify that you are running CCSv4.2 or later and the C6000 code generation tools v7.4 or later (or 7.0.5 for CCSv4). CCS should recognize the installed emulator and prompt you to download the necessary drivers. Once completed, you will then want to setup your DLLView script.

To set up the DLLView script:

1. In the IDE, open the Scripting Console under View > Scripting Console
2. At the Scripting console, type one of the following to load the DLLView script (insert your own source tree path, and make sure to load the version that corresponds to your installed CCS version): Code Composer Studio 4: `loadJSFile "[PATH TO AAX SDK]/TI/CCSv4/dllView_Elf_Avid.js" true` Code Composer Studio 5 and later: `loadJSFile "[PATH TO AAX SDK]/TI/CCSv5/dllView_Elf_Avid.js" true`

You should now see a new menu item under the Scripts menu: "DLLView -Load Pro Tools Plug-In Symbols" This should load every time CCS starts.

12.42.6.4.7 Loading Symbols for Debugging You will need to get your code loaded and running on the TI before you load symbols. You can do this directly through Pro Tools, or by using our DigiShell test tool. If using the DigiShell test tool, load the DAE dish and then a plug-in via the following commands: `load_dish` DAE Loads the DAE dish `run` Lists available plug-ins with their index and `spec run<index>` Instantiates the <index> plug-in

Use the DLLView script to load symbols for ELF DLLs. After setting up the DLLView script and connecting to the desired chip in the Debug pane, run the "DLLView -Load Pro Tools Plug-In Symbols" script from the Scripts menu in Code Composer Studio.

Note

The chip will need to be Suspended in the debugger in order to load symbols.

To load symbols for debugging:

1. In CCS, Launch the TI Debugger (Target > Launch TI Debugger)
2. Connect the debug target to the appropriate chip
3. Suspend the chip
4. Run Scripts > DLLView -Load Pro Tools Plug-In Symbols.

Note

This script can take a moment to load; look at the Scripting Console to view its progress if you like
This script may print a warning about TIShell.out not existing. This warning is benign for plug-in debugging since the TIShell symbols are not required in this case.

This will load symbols for all symbol-rich modules running on the chip(s) connected to the debugger. If you load or unload plug-ins after this, you can simply repeat the "DLLView -Load Pro Tools Plug-In Symbols" command, which will synchronize the debugger with the current configuration.

Note

When running a plug-in in Pro Tools, the first DSP chip is reserved for the HDX mixer. Therefore the first available DSP chip for plug-in instantiation is `C672x_1`. Under DSH, the first available DSP chip is `C672x_0`.

12.42.6.4.8 Breaking on first entry into algorithm To break on the first entry into the plug-in's processing routine, use the manual single-buffer processing mode in DSH: `piproctrigger manual run<index>` Attach debugger, suspend the chip, load symbols, set breakpoint, resume `piproctrigger auto`

12.42.6.4.9 Breaking in the on-chip algorithm initialization callback It is not currently possible to hit a breakpoint in the optional on-chip algorithm initialization callback for a plug-in. If you need to troubleshoot this callback then you should use tracing to print debug information to a log file.

12.42.6.5 Tracing

Avid's AAX DSP platforms provide tracing functionality based on Avid's [DigiTrace](#) tool.

To enable trace logging for TI plug-ins, use the [AAX_TRACE](#) or [AAX_TRACE_RELEASE](#) macros defined in [AAX_Assert.h](#). A separate macro, [AAX_ASSERT](#), is also available for conditional tracing. These macros are cross-platform and will function whether the algorithm is running on the TI or on the host.

12.42.6.5.1 Tracing requirements

- The [AAX_ASSERT](#) and [AAX_TRACE](#) macros are debug-only and will not provide tracing output from release builds of your plug-in. [AAX_TRACE_RELEASE](#) may be used for tracing in both debug and release configurations.
- These macros require that the `DTF__AAXPLUGINS` facility is enabled in the DigiTrace configuration file. You can toggle this facility to enable or disable AAX algorithm-level tracing.
- In order for tracing to be successful on TI platforms, your plug-in's ELF DLL must dynamically link against `TIShell.out`, a component that is installed alongside the Pro Tools application. This file includes the 'glue' that is required in order for the linker to resolve the DigiTrace entrypoint symbol in the DLL.

To link your plug-in project to `TIShell.out` in Code Composer Studio, follow the steps listed in [Linking to TIShell.out](#).

12.42.6.5.2 Tracing example `int32_t` `AAX_CALLBACK`

```
MyExamplePlugIn_AlgorithmInit ( SExample_Alg_Context const *
    inInstance , AAX_EComponentInstanceInitAction inAction )
{
    AAX_TRACE_RELEASE (
        kAAX_Trace_Priority_Normal ,
        "MyExamplePlugIn_AlgorithmInit called for action : %d",
        inAction );
    return 0;
}
```

Listing 2: Adding trace code on TI

12.42.6.5.3 Usage notes

- When running on the DSP, the actual handling of each tracing call occurs in a separate thread. This can lead to incorrect data reporting if volatile data, such as a pointer to an audio sample, is passed in to the tracing statement as a parameter.
- DSP tracing is most reliable when using debug TI builds and when all TI compiler optimizations have been disabled
- Known and resolved issues with DSP tracing are logged on the [Known Issues](#) page

12.42.6.6 Testing in Pro Tools

12.42.6.6.1 The System Usage window The System Usage window in Pro Tools includes some features specifically targeted at testing DSP plug-ins, and particularly for testing shuffle events. Starting in Pro Tools 10, the System Usage window includes the following test features:

- Shift + Drag DSP Meter - This shuffles everything on the chosen chip to another chip, which allows you to quickly test shuffle for a given chip.
- Hover mouse over DSP - Presents a tooltip to show the running plug-ins on a chip
- Cmd+Option+Shift Hover - Detailed debugging tooltip info
- Cmd+Option+Shift Click - Forces a full shuffle of all chips / cards
- Click on empty chip - Reserves a DSP to prevent allocation on that chip

12.42.6.6.2 DSP information tooltip Pro Tools can display additional information for DSP plug-ins using some debug tooltips that are hidden in the plug-in window header and the System Usage window.

The tooltip in the plug-in window header displays information about the particular plug-in instance that is currently shown in the window. To display this tooltip, hold Command-Option-Shift (Mac) or Control-Alt-Shift (Windows) and hover the mouse cursor over the DSP > Native button in the plug-in header.

The tooltip in the System Usage window displays usage information for each DSP chip in the system. You can reveal this tooltip for a particular chip by mousing over the chip's usage meter while holding Command-Option-Shift (Mac) or Control-Alt-Shift (Windows). This tooltip shows the chip's total allocated cycles, internal, and external memory.

The information in these tooltips is generally targeted at systems-level debugging, but can prove useful for some plug-in troubleshooting as well.

Figure 1: DSP tooltip in the Pro Tools plug-in window header.

Figure 2: DSP tooltip in the Pro Tools System Usage window.

12.42.7 Common Issues with TI Development

12.42.7.1 Data structure compatibility

AAX DSP plug-ins use a set of custom data structures to exchange information with host. In order to preserve a consistent binary interface between the plug-in's host and algorithm, the layout of these structures must be identical on both platforms. Each structure must have the same size when compiled by both the host platform compiler and the TI DSP compiler, and any members that are referenced by both the host code and the DSP code must reside at the same offset within the struct on both platforms.

In order to satisfy this requirement, it is essential that an AAX plug-in's algorithm context structure and any other data structures that are passed between the host and the DSP use appropriate alignment. Data structures are usually aligned to 32-bit boundaries, and both Intel and TI compilers use identical struct alignment and packing for most cases. However, this behavior is not explicitly defined in the C standard.

Furthermore, different compilers may use different sizes for some built-in data types. It is therefore very important to use explicitly-sized types such as `int32_t` and `float` rather than ambiguous types such as `bool` or `int`. One particularly tricky data type is pointers, which may be compiled as 64-bit values on a 64-bit Intel system but as 32-bit values on the TI DSP.

Here are some specific scenarios when an unexpected difference in alignment or data type size may occur and cause an ABI incompatibility between a plug-in's host and DSP components:

- [Nested structures](#)
- [Usage of pragma pack](#)
- [Dynamic allocation of memory in structures and algorithm](#)
- [Incorrect use of pointer data](#)
- [Pointer data size incompatibility](#)

12.42.7.1.1 Nested structures It can be particularly difficult to debug alignment issues in nested data structures. One reason is that nested structs do not necessarily have the same alignment as the parent struct. A nested structure will have the alignment that is set preceding its declaration, not the alignment of the structure in which it is contained.

Aside from avoiding nested structs entirely, one way to avoid potential issues is to make sure that nested structs always contain a double. This will guarantee that the structure is double-word aligned. We have also found that placing nested structs near the beginning of the parent struct results in more consistent alignment between Intel and TI compilers, even in cases where the actual alignment of each member is strictly ambiguous according to the standard.

Another important rule of thumb with nested structs is to define them inline in the enclosing structure. We have found that including one data structure as a member in another data structure will only be reliably aligned between Visual Studio and the TI compiler tools if the member structure's type is defined in-line. This does not appear to be an issue between clang and the TI compiler - the data structure alignment for the nested structure is consistent between those two compilers regardless of the location of the internal structure's definition.

```
#include AAX_ALIGN_FILE_ALG
struct SomeStruct
{
    float a;
    float b;
};
#include AAX_ALIGN_FILE_RESET

// Somewhere else...
#include AAX_ALIGN_FILE_ALG
class SomeClass
{
public:
    SomeStruct s; // Don't do this! Inconsistent between Visual Studio and TI

    // other stuff...
};
#include AAX_ALIGN_FILE_RESET
```

Listing 3: Problematic code: nested struct not defined in-line

```
#include AAX_ALIGN_FILE_ALG
class SomeClass
{
public:
    struct SomeStruct
    {
        float a;
        float b;
    } s; // This is fine - consistent between Visual Studio, clang, and TI

    // other stuff...
};
#include AAX_ALIGN_FILE_RESET
```

Listing 4: Fixed code: nested struct defined in-line

12.42.7.1.2 Usage of pragma pack If you use pragmas to align your structs, then you should know that in most cases it will only decrease the natural struct alignment of a compiler. That means that if you have

```
#pragma pack(8)
struct x
{
    char a;
    float b;
};
```

Listing 5: Example of usage of #pragma pack where it has no effect

then struct x most likely won't be aligned to the 8 byte boundary. Therefore the pack pragma is not really useful for addressing alignment issues. Instead of using pack, one way to guarantee that a structure is double-word aligned, is to include at least one double member.

```
#pragma pack(8)
struct x
{
    float a;
    double b;
};
```

Listing 6: Example of usage of #pragma pack where it actually affects the alignment of the structure

In this case data will be double-word aligned.

12.42.7.1.3 Dynamic allocation of memory in structures and algorithm The problem with dynamic allocation is that it's difficult to enforce specific alignment of the resulting block beyond the natural alignment of the structure. Newly allocated blocks are not double-word aligned by default. This prevents double-word memory access optimizations (see [Additional data type optimizations](#)) from working.

```
// blocks are not aligned to 8-byte boundaries by default. This prevents double-word
// memory access optimizations from working
float* floatBlock = new float[100];
delete[] floatBlock;

// Though AAX_Alignment.h does include some aligned memory allocators to counteract the alignment
// problem, their use is still strongly discouraged.
float* floatBlock2 = alignMalloc<float>(100, 8);
alignFree(floatBlock2);
```

Listing 7: Problems which may arise when using dynamic allocation of memory in algorithm

12.42.7.1.4 Incorrect use of pointer data In general, you should avoid storing pointers to anything in any data structures that are passed between the host and the DSP. There are many possible problems and bugs that can be caused by this, for example:

- Often the memory map of packets can change out from under the plug-in
- It is easy to accidentally reference data in the wrong memory space when setting pointer values
- Pointer data types are not explicitly sized (see [below](#).)

One alternative to using raw data pointers is to store data offsets into a coefficient array rather than using direct pointers to other structure elements. A solution such as this that does not involve pointer data types will almost always end up being easier to implement, easier to troubleshoot, and easier to maintain than a solution that uses pointer data.

That said, if you must use pointer data types in any data structures that are passed between the AAX host and DSP components then you should be very careful to avoid the problems listed above.

12.42.7.1.5 Pointer data size incompatibility Problems due to pointer data size incompatibility can be particularly difficult to debug. Pointer data types are not explicitly sized in C, and, starting with the 64-bit Pro Tools 11 release, pointers will have different lengths for host and TI binaries. This can cause subtle portability problems in certain circumstances, if proper care is not taken.

Consider the following state block:

```
struct SMyPlugInStateBlock
{
    float mInGain_Smoothed;
    some_t* mPointerP;
    float mOutGain_Smoothed;
};
```

Notice the pointer `mPointerP` (the type that it points to is irrelevant for this discussion). Perhaps it is a pointer that can reference different sets of coefficients, or perhaps it points to some sort of global variable. In any case, this pointer is 64-bits long on the host, and 32-bits long on TI.

In most cases, this won't cause a problem because the host simply allocates a bit more space for the state block than the TI needs and fills the allocated memory with 0s. But consider the case where we overload `ResetFieldData()` to set `mOutGain_Smoothed` to something other than 0:

```
AAX_Result MyPlugIn_Parameters::ResetFieldData (AAX_CFieldIndex inFieldIndex, void * inData, uint32_t
inDataSize) const
{
    AAX_Result result;
    switch (inFieldIndex)
    {
        case (eMyAlgFieldIndex_State):
        {
            memset(inData, 0, inDataSize);
            SMyPlugInStateBlock* stateP = static_cast<SMyPlugInStateBlock*>(inData);
```



```

        stateP->mOutGain_Smoothed = mOutGain_Target;
        result = AAX_SUCCESS;
        break;
    }
    default:
    {
        result = AAX_CEffectParameters::ResetFieldData(inFieldIndex, inData, inDataSize);
        break;
    }
}
return result;
}

```

We might be doing this if `mOutGain_Smoothed` was a smoothing parameter and we want to start it at the target gain value (rather than having it smooth from 0.0 at instantiation). But if the Host and TI can't agree on where in the state block `mOutGain_Smooth` is located, then the result will be unexpected behavior that is difficult to debug.

The most direct way to avoid this problem is to use an explicitly-sized 32-bit type for any pointers in your state block:

```

struct SMyPlugInStateBlock
{
    float mInGain_Smoothed;
    uint32_t mPointerP;
    float mOutGain_Smoothed;
};

```

It will be necessary to use `reinterpret_cast<float*>(stateP->mPointerP)` to recast the pointer to a pointer data type on the TI, but that should not result in any extra processing cycles.

12.42.7.1.6 Alignment Reference These are the data type sizes and default alignments for some common compilers when compiling for 64-bit binary formats:

	TI		MS Visual C++		C++ Builder		GCC	
char	1 byte	1-byte aligned	1 byte	1-byte aligned	1 byte	1-byte aligned	1 byte	1-byte aligned
short	2 bytes	2-byte aligned	2 bytes	2-byte aligned	2 bytes	2-byte aligned	2 bytes	2-byte aligned
int	4 bytes	4-byte aligned	4 bytes	4-byte aligned	4 bytes	4-byte aligned	4 bytes	4-byte aligned
long	4 bytes	4-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned
long long	8 bytes	8-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned
bool	1 byte	1-byte aligned	1 byte	1-byte aligned	1 byte	1-byte aligned	1 byte	1-byte aligned
float	4 bytes	4-byte aligned	4 bytes	4-byte aligned	4 bytes	4-byte aligned	4 bytes	4-byte aligned
double	8 bytes	8-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned
long double	8 bytes	8-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned	16 bytes	16-byte aligned
pointer	4 bytes	4-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned	8 bytes	8-byte aligned

Also here are some useful links to web resources on the topic:

- A good resource for the TI DSP is <http://www.ti.com/lit/an/sprab89/sprab89.pdf> (Section 2 especially). This document includes some graphs of simple alignment examples.
- Another good reference regarding general struct alignment issues is available from [publib.boulder.ibm.com](http://publib.boulder.ibm.com/infocenter/macxhelp/v6v81/index.jsp?topic=/com.ibm.vacpp6m.doc/compiler/ref/rnpgpack.htm): <http://publib.boulder.ibm.com/infocenter/macxhelp/v6v81/index.jsp?topic=/com.ibm.vacpp6m.doc/compiler/ref/rnpgpack.htm>

12.42.8 TI Optimization Guide

Optimizing AAX real-time algorithms for Avid's TI-based platforms is very similar to optimizing real-time algorithms for any architecture. When developers think about optimization, they often think "I want to make my code run faster". In reality, however, optimization is about making the processor do less. After all, the processor's clock rate is fixed and can only perform a limited number of instructions in a set amount of time. Therefore, our focus in this section will be on helping the compiler produce code with shorter execution paths and make full use of the TI chip's architecture.

Modern compilers have become extremely powerful at being able to optimize code, which is fortunate given the complicated architectures of today's DSP products. In this section we will not focus on instruction-level "optimizations" like the one below, which will automatically be done by the compiler. Instead of making our code faster, which it won't, little "tricks" like this really just make code harder to read:

```
int y = x;
y = y >> 1; // y = y / 2;
```

Listing 8: The kind of optimization that you won't be seeing in this section

Rather, we will focus on refactoring audio processing algorithms to be more efficient and on giving the TI compiler better information about the code, pointers, and data it is working with so it can perform more effective compile-time optimizations.

Finally, our optimization efforts will focus on the worst-case code path. For example, developers often try to optimize algorithms by conditionally bypassing portions of code that may be disabled by particular parameter states. This is counter-productive, because the system has to assume a plug-in's worst-case execution performance regardless of how much time the plug-in is actually using. Therefore, in the context of real-time algorithms running on AAX DSP platforms, it is best to only worry about worst-case execution time.

For more information about using TI's toolset to profile your code's performance, see [Cycle count performance test](#).

Note

The optimizations described in this section assume that you are using version 7 or higher of TI's C6000 Code Generation Tools (CGTools). We strongly recommend using v7.0.5 or later as earlier versions throw linking errors.

12.42.8.1 Optimization quick start

Here is a quick outline of the general optimization steps for an AAX DSP algorithm:

1. Before beginning your DSP optimizations, make sure that your Native algorithm has basic optimizations in place. In our experience, beginning the TI optimization process with a slow or needlessly precise Native algorithm will result in a long porting process. Here are some suggestions for common Native optimizations:
 - Identify unnecessary double precision
 - Identify tables that have too high of granularity
2. Make sure your compiler Release settings enable the compiler to optimize fully and give full optimization comments: `-k -s -pm -op3 -os -o3 -mo -mw -consultant -verbose -mv67p`
3. Use the load/update/store design pattern to reduce memory accesses in inner loops
4. Move any processing that does not directly depend on the audio signal out of the real-time algorithm
5. Declare non-changing variables and pointers (both local and in parameter lists) as `const`
6. Declare non-aliased pointers (both local variables and function parameters) as `AAX_RESTRICT`
7. Change any `long` variables to `int`, and change `double` variables to `float` if the reduced precision does not affect signal integrity (usually defined as cancellation with the plug-in's Native algorithm.)
8. Restructure inner processing loops so that they do not contain large conditional statements or other branches
9. Declare any functions that are called within the innermost processing loop as `inline` in order to allow the inner loops to pipeline
10. Add loop count information when known, using `#pragma MUST_ITERATE(min,max,quant)`

12.42.8.2 Compiler and linker options

As with any complex environment, many performance gains on the TI rely on the appropriate compiler and linker options. The options documented here will allow CGTools to apply its optimization logic to your algorithm.

When tweaking compiler options on the TI, keep in mind that, like on any CPU, it is useless to optimize Debug code or to profile its performance. This is especially true on TI processors because of the fact that generated Debug and Release assembly is almost completely different, assuming that heavy optimization options were chosen for the Release configuration.

In general, all recommended compiler options should be set correctly in the AAX SDK's example plug-in projects, and these settings may be used as a guide for your own plug-in projects. See the SDK files `CommonPlugIn_↔CompilerCmd.cmd` and `CommonPlugIn_LinkCmd.cmd` for the latest recommended settings.

12.42.8.2.1 Overview of optimization-related compiler options

- `-g` Full symbolic debug. This setting should be used in debug configurations to make stepping through code easier. It should not be defined in release configurations, as it will prevent the compiler from being able to fully optimize code.
- `-k` Keep generated .asm files. This should be turned on in release configurations so that you can use the ASM output as feedback when making optimization decisions and performance improvements.
- `-d "_DEBUG"` Defines the `_DEBUG` preprocessor macro that alters how certain code is generated (asserts, stdlib, etc). This should be turned on in debug configurations only. Note that TI does not require `NDEBUG` to be defined in release configurations.

Note

This will eventually be deprecated in favor of the pre-defined `"_TMS320C6X"` macro.

- `-mv67p` Specifies that the compiler should build code for the C67x+ chip variant we are using, which has some improvements beyond the original C67x. This option should be enabled in all build configurations that target the HDX platform.
- `-s` Specifies Opt-C/ASM interlisting. This interweaves modified C-code and ASM in the .ASM file produced by the `-k` option. You should use `-s` in release configurations so that the ASM file can be read more easily.

Note

Do NOT use the `-ss` option in release configurations. This option will negatively affect optimization

- `-pm` Program mode compilation. Instructs the C compiler to compile all files in the same compilation unit, so that it can optimize code further using information from all files being compiled. See [Program Mode optimization \(-pm\)](#) for more information.
- `-op3` A modifier for the `-pm` option, this specifies that there are no external variable references in the project. This option is appropriate for TI algorithms, which do have an external function reference (the process entry point) but do not have external variable references. This option allows the compiler to further optimize global variables without worrying whether they will be accessed outside of the compilation unit. See [Program Mode optimization \(-pm\)](#) for more information
- `-o3` File-level optimization. This flag gives the compiler full ability to optimize C-code by reordering instructions, inlining functions, and performing other optimizations. Note that the resulting ASM code will be very difficult to parse back into the original C and will make debugging very difficult, so this flag should only be used for Release code. See [Optimization flags \(-o\)](#) for more information.
- `-mo` Use Function Subsections. This instructs the compiler to place all functions into their own separate subsection in the linker map. This allows the linker to remove unused functions in order to reduce memory usage.

- `-mw` Generate a single iteration view of SP loops. This flag adds important information to the ASM output file that is useful when optimizing your code for pipelined loops.
- `-verbose` Output verbose status messages when compiling files. Though not very useful for humans, verbose output will produce some key information that text parsers can use, such as compiler versions and other details.

12.42.8.2.2 Overview of optimization-related linker options

- `-relocatable` Generate a relocatable non-executable.
- `-m"file.map"` Generate a map file. This file contains useful information about the memory footprint of your plug-in, which is useful for fixing large plug-ins that may not have fit into available program memory.
- `-w` Warn about output sections. This flag generates very useful information that tells you if there might be a problem with memory output sections you are trying to generate.
- `-x` Exhaustively read libraries. This is a useful flag if you do not want to worry about the order in which you specify required libraries.

12.42.8.2.3 Optimization flags (-o)

- Register (`-o0`) This option allows for some performance gains over non-optimized code by allocating variables to registers, inlining functions declared inline, etc.
- Local (`-o1`) This option enables local optimizations, with very similar results to the register-level optimizations of `-o0`.
- Function (`-o2`) This is the standard optimization level, and provides large gains over unoptimized code. This optimization level allows function-level optimizations such as software pipelining, loop optimization/unrolling, etc.
- File (`-o3`) This option can provide some speedup beyond function-level optimizations, but also mutilates assembly code beyond recognition. At this optimization level the compiler will remove unused functions, simplify code in the case of unused return values, auto-inline small functions, etc.

Like the corresponding Visual Studio options, `-o0` and `-o1` allow you to step through code line-by-line for debugging, at the cost of reduced performance. `-o2` and `-o3` sacrifice the ability to step through code and watch memory in favor of optimized code.

12.42.8.2.4 Program Mode optimization (-pm) Program mode optimization gives the compiler further optimization information by compiling all files at once rather than individually. Thus global constants, function implementations, etc. can be made known to the entire program at compilation. This allows the compiler to inline functions more effectively and to determine loop unrolling based on constant loop iterators.

There are a few `-pm` options:

- `-pm -op0` Contains functions and variables that are called or modified from outside the source code provided to the compiler.
- `-pm -op1` Contains variables modified from outside the source code provided to the compiler but does not use functions called from outside the source code.

This option is not appropriate for AAX plug-in algorithms, because the algorithm component will be exported and called from outside the compiled source code.

- `-pm -op2` Contains no functions or variables that are called or modified from outside the source code provided to the compiler.

This option is not appropriate for AAX plug-in algorithms, because the algorithm component will be exported and called from outside the compiled source code.

- `-pm -op3` Contains functions that are called from outside the source code provided to the compiler but does not use variables modified from outside the source code.

This is the recommended Program Mode optimization level for TI plug-ins. This optimization level requires that no global variables are used outside of the algorithm callback. In general, any such variables should be passed in to a TI algorithm via the algorithm's context structure.

12.42.8.2.5 Compiler options to avoid The following information was taken from the TMS320C6000 Programmer's Guide:

- `-g/-s/-ss` These options limit the amount of optimization across C statements, leading to larger code size and slower program execution.
- `-mu` This option disables software pipelining for debugging. If a reduction in code size is necessary, use the `-ms2/-ms3` options. These options will disable software pipelining among their other code size optimizations.
- `-mz` This option is obsolete. When using 3.00+ compilers, this option will decrease performance and increase code size.

12.42.8.3 The load-update-store pattern

The load-update-store pattern is one of the cornerstones of a fast iterative algorithm. This pattern specifies that locally accessed data should be loaded into memory at the start of processing, accessed during processing, and stored or saved after processing has completed. By using this pattern you will move memory reads and writes outside of your plug-in's innermost processing loop, which reduces data dependencies and shortens the critical inner loop.

As an example, consider the following unoptimized filter code:

```
inline void
ProcessDirectFormII(float* input, float* output, float* state, float*
    coefs, int nsamp)
{
    // eB0 .. eB2 and eA0, eA1 are just integer enums to partition
    // the filter coefficients into A and B
    for(int i = 0; i < nsamp; ++i)
    {
        output[i] = input[i]*coefs[eB0] + state[0];
        state[0] = input[i]*coefs[eB1] + state[1] - output[i]*coefs[eA0];
        state[1] = input[i]*coefs[eB2] - output[i]*coefs[eA1];
    }
}
```

Listing 9: Unoptimized filter algorithm

Notice that in this code there are at least 15 memory accesses per loop iteration! This algorithm will be very inefficient as the value of `nsamp` increases.

The compiler should be able to optimize this algorithm to some extent by pulling certain memory accesses outside of the loop. However, the compiler cannot completely optimize the loop because it must assume that the input/output/state/coefs pointers are aliased in memory. We will discuss the `const` and `restrict` keywords later, which are ways to give the compiler additional information it can use to optimize this loop. However, for now let's focus back on the basic design of this code.

Using load-update-store, we can refactor this loop to pull the memory accesses outside of the loop:

```
void
ProcessDirectFormII (float* input, float* output, float* state, float *
    coefs, int nsamp)
```

```

{
    // eB0 .. eB2 and eA0, eA1 are just integer enums to partition
    // the filter coefficients into A and B

    // ---- LOAD ----
    float coefA0 = coefs [eA0];
    float coefA1 = coefs [eA1];
    float coefB0 = coefs [eB0];
    float coefB1 = coefs [eB1];
    float coefB2 = coefs [eB2];

    float state0 = state [0];
    float state1 = state [1];

    float output;

    // ---- UPDATE ----
    for (int i = 0; i < nsamp; ++i)
    {
        output = input [i]* coefB0 + state0;
        state0 = input [i]* coefB1 + state1 - output * coefA0;
        state1 = input [i]* coefB2 - output * coefA1;
        output [i] = output;
    }

    // ---- STORE ----
    state [0] = state0;
    state [1] = state1;
}

```

Listing 10: Refactored filter algorithm with load-update-store pattern applied. Not fully optimized.

Though the code initially appears longer, you will notice that we have reduced the loop to only 4 memory accesses! Though we have an additional 9 memory accesses outside the loop, they will only occur once per function call, resulting in significant savings at higher values of `nsamp`.

Note

we are not finished with this loop yet, because we can make some very significant gains by using the `restrict` and `const` keywords, as discussed in the section on [C keywords](#).

Before moving on from load-update-store, let's consider how this pattern should be applied to different categories of data that may be provided in an AAX DSP processing context:

- **Coefficients** and **parameters** are read-only by definition. As such, they should be loaded into a local variable at the beginning of the algorithm callback and should not be modified further.
- **Private state** State parameters are writable and may be changed by the algorithm. Therefore, private state data should be loaded into a local variable copy, then stored back into memory after the local copy is updated.
- **Output** Output is write-only, so all calculations may be performed on a local variable and then stored into memory once per loop.

12.42.8.4 Case study: IIR filter implementation on TI 672x DSPs

In this section we will examine various IIR filter implementations as a specific example of the considerations that must be made when optimizing DSP code for the 672x.

The TI 67xx family of DSPs is notably different from some other typical DSP processors, such as the 56k and the Intel FPU, in that the TI DSP does not have an implicit higher-precision multiply-accumulate. It is of course capable of double precision accumulation, but this must be coded explicitly. In some ways, this is similar to the Intel SSE processing unit, which jetisonned the 80-bit floating point stack used in the Intel FPU. The lack of higher precision accumulation in TI (and SSE) can sometimes result in unacceptable quantization noise performance for single precision filter implementations. Luckily, with the right choice of filter structure or coding for explicit double precision accumulation, excellent results can be achieved.

On fixed-point DSPs such as 56k, Direct Form I (DF1) implementation is the standard due to moderately good fixed point scaling properties, decent noise performance, and simple implementation. However, on a 672x DSP a single precision DF1 filter can have terrible noise performance (depending on the filter coefficients and the audio material being processed.) A degenerate case is a DF1 highpass filter processing low frequency material; in DF1, the feedforward coefficients subtract the previous sample from the current sample, and for low frequency material this produces very small numbers with low precision. Single precision DF2 structures also produce similarly poor results in this respect.

One option to improve upon these results is to use double precision throughout the 672x filter implementation. However, this results in a heavy cycle performance penalty due to the high cost of double operations on the TI DSP. Another, often better, option is to use single precision coefficients and state, with double precision accumulation:

```
float in, b0, b1, a1, state1;
double accum;
accum = double (b0) * double (in) +
        double (b1) * double (state1) +
        double (a1) * double (accum);
state1 = in;
```

Listing 11: Mixed-precision DF1 filter implementation

The TI compiler will implement this using the mpysp2dp instruction, since it knows that the operands started out as single precision and end up as double precision. This is considerably faster than going to a full double precision implementation, but it is still relatively slow compared to straight single precision. Making the state double precision will improve noise performance further, with some increase in cycle usage.

Another option that generally gets good results is the single precision DF2 Transpose (DF2T) filter. On TI the DF2T implementation is fast and generally has good noise performance. If you are looking for a simple recommendation that should work well enough for most applications, DF2T is a good choice.

The optimized C filter library available from TI uses the DF2 structure in its implementation. Even though DF2 has some limitations, this is a good starting point for seeing how to optimize filter code on TI; peak performance on TI is 2.25 cycles per biquad, so it's pretty amazing what can be done (to achieve that level of performance multiple series or parallel biquads need be put in a tight loop.) We have adapted some of this filter code to DF2T, and still achieved fairly similar cycle performance.

If the single precision DF2T noise performance is not good enough for your application, then either double precision or one of the myriad other filter structures, such as State Space, Gold-Rader, Lattice or Zolzer, should do the job. In fact, there is one relatively new filter structure which we think stands out, called the Direct Wave Form (DWF) filter. Details about this filter structure can be found in *Direct Wave Form Digital Filter Structure: an Easy Alternative for the Direct Form* by Jean H.F. Ritzerfel. According to the author the noise performance is 3dB within optimal, it's relatively efficient (5 multiplies per biquad), free of limit cycles, has simple coefficient generation and low coefficient quantization sensitivity. It might just be the perfect filter structure, but we'll let you be the judge of that; keep in mind that all filter structures have some tradeoffs, and the recommendations made here might not be the best for your particular application.

12.42.8.5 Understanding CGTools-generated ASM files

The ability to read the ASM files that are generated by CGTools is essential when optimizing a TI algorithm. Specifically, the information in these files will allow you to determine if anything is preventing software pipelining from occurring, which is the single most effective form of optimization on the C6727.

To view your project's ASM file, turn on the `-k` compiler option ("Keep Generated .asm Files", found under Build Options > Compiler > Assembly in the Code Composer Studio IDE.) By default, ASM files will be placed in the same directory as the corresponding source file.

Note

You should only examine ASM listings of Release code that has been optimized by the compiler. Debug code should not be optimized.

Each ASM file for a TI algorithm callback should contain text that marks the start of the assembly listing for the processing loop. For example:

```
*****
; * FUNCTION NAME: // [Your algorithm's ProcessProc symbol] _____ *
; * _____ *
; * Regs Modified: A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14, _ *
; * _____ *
; * _____ B13,SP,A16,A17,A18,A19,A20,A21,A22,A23,A24,A25, _____ *
; * _____ A26,A27,A28,A29,A30,A31,B16,B17,B18,B19,B20,B21, _____ *
; * _____ B22,B23,B24,B25,B26,B27,B28,B29,B30, B31 _____ *
; * Regs Used____: A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14, _ *
; * _____ A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12, _____ *
; * _____ B13,DP,SP,A16,A17,A18,A19,A20,A21,A22,A23,A24, _____ *
; * _____ A25,A26,A27,A28,A29,A30,A31,B16,B17,B18,B19,B20, _____ *
; * _____ B21,B22,B23,B24,B25,B26,B27,B28,B29,B30,B31 _____ *
; * Local Frame Size: 0 Args + 148 Auto + 44 Save = 192 byte _____ *
*****
```

Listing 12: CGTools-generated header for a processing loop assembly listing

Within this listing, you are looking for several things:

1. Function calls
2. Branches or control code
3. Software pipelining notes

12.42.8.5.1 Function calls [!B0] CALL .S1 __divd ; [213]
|| [!B0] MVKH .S2 0x40080000 ,B5 ; [213]
|| [B0] MV .L1X B10 ,A4 ; [213]
\$C\$RL9 : ; CALL OCCURS {__divd} ; [213]

Listing 13: Function call in a CGTools-generated assembly listing

Function calls, such as the call in the listing above, cannot be effectively pipelined. If you find a function call figure out what C instruction it is caused by. Sometimes a function call will be made implicitly, such as when casting from float to int or when doing division. All function calls should be removed from the processing loop or inlined in order for the compiler to optimize effectively.

12.42.8.5.2 Branches NOP 1
B .S1 \$C\$L5 ; [213]
NOP 4
MPYDP .M1X A5:A4 ,B5:B4 ,A11:A10 ; [213]
|| LDW .D2T2 ** SP (124) ,B5 ; [218]
; BRANCH OCCURS { \$C\$L5 } ; [213]

Listing 14: Branch in a CGTools-generated assembly listing

Branches can also prevent loop pipelining. If you find a branch in your algorithm's assembly, determine whether it is preventing the compiler from pipelining a loop. If it is preventing pipelining, you must figure out how to rewrite the conditional in your C code so that it will not be compiled into a branch.

12.42.8.5.3 Software pipelining notes For each loop the compiler finds and is able to pipeline, the .ASM file should contain a section similar to the one below:

```

/*-----*
/* SOFTWARE PIPELINE INFORMATION
/*
/* Loop source line : 68
/* Loop opening brace source line : 69
/* Loop closing brace source line : 124
/* Loop Unroll Multiple : 2x
/* Known Minimum Trip Count : 1
/* Known Max Trip Count Factor : 1
/* Loop Carried Dependency Bound (^) : 15
/* Unpartitioned Resource Bound : 20
/* Partitioned Resource Bound (*) : 20
/* Resource Partition :
/* A- side B- side
/* .L units 0 0
/* .S units 0 1
/* .D units 20* 20*
/* .M units 7 5
/* .X cross paths 5 6
/* .T address paths 20* 20*
/* Long read paths 5 1
/* Long write paths 0 0
/* Logical ops (. LS) 5 4 (.L or .S unit )
/* Addition ops (. LSD) 0 1 (.L or .S or .D unit )
/* Bound (.L .S .LS) 3 3
/* Bound (.L .S .D .LS .LSD) 9 9
/*
/* Searching for software pipeline schedule at ...
/* ii = 20 Schedule found with 3 iterations in parallel

```

Listing 15: Pipelined loop header in a CGTools-generated assembly listing

These are the important items to note in this listing:

- **Loop Carried Dependency Bound and Partitioned Resource Bound** The maximum of these numbers is the minimum number of clock cycles one instance of the loop will require in its current form. You can reduce these numbers by performing some of the optimizations listed in this guide.
- **Loop Unroll Multiple** This line will appear if the compiler is partially unrolling the loop to improve performance.

If a loop section instead displays `Disqualified loop`: then some of the conditions required to enable software pipelining have not been met:

- `-o2` or `-o3` optimizations must be enabled
- The loop cannot contain a function call. Make all called functions inline.
- The loop cannot contain any branches or jumps, often caused by large conditional statements
- Software pipelining will not work with nested loops; only the innermost loop will be pipelined. You should completely unroll the inner loop or refactor the algorithm so that the loop can be pipelined

For more information about pipelining and loop/branch optimization, see [Refactoring conditionals and branches](#).

12.42.8.6 C keywords

There are a few keywords in C that give the compiler additional information about the variables you declare and parameters you pass into functions. This allows the compiler to further optimize the code it is compiling, which can result in significant performance gains.

12.42.8.6.1 const Effective use of `const` lets the compiler know whether pointers, scalars, or objects will remain constant in memory.

Let's add the `const` keyword to the filter function from our example of [The load-update-store pattern](#).

```
void
ProcessDirectFormII (
    const float * const input, // read - only
    float * const output, // read - write
    float * const state, // read - write
    const float * const coefs, // read - only
    int nsamp )
{
    // eB0 .. eB2 and eA0, eA1 are just integer enums to partition
    // the filter coefficients into A and B

    // ---- LOAD ----
    const float coefA0 = coefs [ eA0 ];
    const float coefA1 = coefs [ eA1 ];
    const float coefB0 = coefs [ eB0 ];
    const float coefB1 = coefs [ eB1 ];
    const float coefB2 = coefs [ eB2 ];

    float state0 = state [0];
    float state1 = state [1];

    // ---- UPDATE ----
    for (int i =0; i< nsamp; ++i)
    {
        const float output = input [i]* coefB0 + state0 ;
        state0 = input [i]* coefB1 + state1 - output * coefA0 ;
        state1 = input [i]* coefB2 - output * coefA1;
        output [i] = output;
    }

    // ---- STORE ----
    state [0] = state0;
    state [1] = state1;
}
```

Listing 16: Refactored filter algorithm with load-update-store pattern and `const` keyword applied.

It is especially important to note that the declaration of `const float output` was moved inside the loop. Why did we do this? Because we see that `output` is constant over an iteration of the loop, but it does change between iterations. By declaring it `const` inside the loop body we remove the data dependency that existed in `output` and allow the loop to optimize more effectively.

As demonstrated by this change to `const float output`, `const` is useful for manually breaking dependencies in DSP code. Variable re-use introduces unnecessary data dependencies in code, which can be avoided by using individual local `const` variables.

12.42.8.6.2 restrict The `restrict` keyword tells the compiler that a specific pointer is not aliased, meaning that none of the memory locations accessed by the pointer are read or written to by any other variable within its local scope. This keyword is very important when optimizing TI code that involves pointers, as all AAX algorithms do due to the nature of the algorithm context structure.

`restrict` was introduced with the C99 standard. AAX plug-ins use the `AAX_RESTRICT` keyword, which is a cross-platform macro for the C99 standard `restrict`.

Note

Now that MSVC has added C99 support to its compiler, `AAX_RESTRICT` will eventually be deprecated in favor of the `restrict` keyword.

The following example demonstrates the use of `restrict` in our filter code.

```
void
ProcessDirectFormII (
    const float * const AAX_RESTRICT input,
    float * const AAX_RESTRICT output,
    float * const AAX_RESTRICT state,
    const float * const AAX_RESTRICT coefs ,
```

```

int nsamp )
{
    // eB0 .. eB2 and eA0, eA1 are just integer enums to partition
    // the filter coefficients into A and B

    // ---- LOAD ----
    const float coefA0 = coefs [ eA0 ];
    const float coefA1 = coefs [ eA1 ];
    const float coefB0 = coefs [ eB0 ];
    const float coefB1 = coefs [ eB1 ];
    const float coefB2 = coefs [ eB2 ];

    float state0 = state [0];
    float state1 = state [1];

    // ---- UPDATE ----
    for (int i =0; i<nsamp; ++i)
    {
        const float output = input [i]* coefB0 + state0;
        state0 = input [i]* coefB1 + state1 - output * coefA0;
        state1 = input [i]* coefB2 - output * coefA1;
        output [i] = output;
    }

    // ---- STORE ----
    state [0] = state0;
    state [1] = state1;
}

```

Listing 17: Refactored filter algorithm with load-update-store pattern and const and restrict keywords applied.

Note

- This example applies `restrict` to the algorithm's input and output audio buffer pointers. These pointers do not alias each other in most algorithms, but this may not be the case for all algorithms and should be verified by the developer before applying `restrict`.
- The `restrict` keyword is somewhat redundant when used with the load-update-store pattern. This is because by asserting to the compiler that the pointers are not aliased, it should be able to partially do the load-update-store refactoring automatically. However, because some compilers have limited or no support for the `restrict` keyword, using the load-update-store pattern is still recommended.

12.42.8.6.3 Keywords to avoid There are some keywords which do more harm than good, but are still being used either due to legacy code or developer superstitions. These keywords should not be used in AAX plug-ins.

- `register` The `register` keyword is a suggestion to the compiler that a certain variable will be accessed frequently and should be stored in a register rather than a memory location. Use this keyword only when you are sure that the compiler is placing a frequently-used variable in memory when it would be advantageous to keep it in a register. Note that the `register` keyword has no effect if the CGTools optimizations are enabled.
- `static` In C, the `static` keyword tells the compiler to initialize the variable at compilation time and retain the value between calls. Though there are some valid situations to use the `static` keyword, its use in AAX plug-ins on all platforms is extremely limited. One of its most "popular" uses, declaring local variables inside a function as `static` in order to achieve a type of global counter, should never be used in AAX algorithm code. If you are using `static` to make a local variable hold its variable across calls to a function, it is always preferable to either pass it in to the function as a modifiable parameter or declare it as a member variable of the method (if C++).

12.42.8.7 Data types

The TI C672x+ is a 32-bit floating point DSP platform, and has a few peculiarities that you should be aware of.

- Use `int` instead of `long` Integers of type `long int` are 40 bits wide on TI, and are very inefficient. Always use the `int` data type (or, even better, the C99-standard `int32_t`) instead.

- Use `float` instead of `double` Double-precision floating-point data types have a significant performance penalty on TI processors. Use `float` instead of `double` wherever possible, as long as this substitution does not affect signal integrity or cancellation.
- Use unsigned values when referencing memory In general, explicitly typed pointers should always be used to reference memory. If you do have need of a generic memory representation, use an unsigned integer to avoid implicit conversion costs.

12.42.8.7.1 Unintended data type conversions When developing for the TI platform it is important to keep an eye out for unintended type conversions, and especially for implicit double-precision instructions. The following points are helpful for both program efficiency and for future maintenance of the code, since they clarify the developer's understanding of how the code should operate, e.g. by specifying that a cast is occurring, and make it obvious that steps such as data type conversions are an intentional part of the algorithm.

- Explicitly declare constants as single-precision. For example, use `0.0f` instead of `0.0`. Often a compiler will be able to do this automatically at compile time, but it is better to be explicit with your intended precision.
- If any casts are required in your code, make them explicit. For example, `float output = (float)doubleVar` as opposed to `float output = doubleVar`.
- Use single-precision `math.h` functions (such as `fabsf()`) instead of the double-precision equivalents (`fabs()`).
- Do not directly reference memory addresses using integer data types; instead, use a pointer data type. If an integer data type is required, use an unsigned 32-bit type.

To help ensure that you are not violating these principles, always be aware of any warnings generated by the compiler. In particular, do not ignore warnings related to "implicit conversion from 'double' to 'float'" or "implicit conversion from 'double' to 'int'"; these warnings may indicate that you are declaring a double when a float would be just as good.

In the final stages of optimization, examine the generated assembly code to make sure there are no unintended double-precision instructions or memory accesses.

12.42.8.7.2 Additional data type optimizations The AAX SDK includes cross-platform macros that can be used to convert two single-precision float loads to one double-precision load. The coefficient smoothing case study below includes an example use case for these macros.

```
const float * pTable = &SmoothCoefTable[address];
AAX_ALIGNMENT_HINT(pTable, 8);
float firstCoef = AAX_LO(*pTable);
float secondCoef = AAX_HI(*pTable);
```

Listing 18: Example of using AAX macros for converting two `float` loads to one `double` load.

In this example the `AAX_ALIGNMENT_HINT` macro checks whether data is aligned on a 8-byte boundary, then the double word is loaded, and finally the `AAX_LO` and `AAX_HI` macros get the double word's first and second (`float`) parts.

If `SmoothCoefTable` consists of floats and is 8-byte aligned, then this scenario will work fine for loads when `address` is even. This raises the question about how to load double word from `&SmoothCoefTable[address]`, when `address` is odd. Since this kind of optimization is most useful for loading data from external memory, where the CPU savings of a single double word load vs two 32-bit loads is greatest, then one trick which can help is to trade off memory (as external memory is plentiful) for performance. Specifically, `SmoothCoefTable` can be organized in a such way that for every member of this table, except the first and the last ones, there will be two consequent entries.

```
const int32_t size = 4;
// instead of this classic variant...
const float SmoothCoefTable[size] = {
    -0.1, -0.2, -0.3, -0.4
```

```

}

// ...table can be organized this way
const float SmoothCoeftable[size*2 - 2] = {
    -0.1, -0.2,
    -0.2, -0.3,
    -0.3, -0.4,
    -0.4, 0.0 /* last member is dummy */
}

```

Listing 19: Example of restructuring the table so that it can be easily used in the optimization scenario given above.

In this case the number of loads will be halved at the cost of doubling the size of the table. If the table is located in external memory then the additional memory requirement can be an excellent trade-off for the performance gained.

12.42.8.8 Case study: Efficient parameter smoothing at single and double precision

Coefficient smoothing ("de-zippering") can often be one of the most difficult parts of a plug-in to optimize for real-time operation. This is especially true in cases when full double-precision smoothing filters have been used in a plug-in's Native code, with the possibility of very small coefficients. In these cases it can be difficult to optimize the smoothing code while also satisfying requirements for audio data parity between the plug-in's Native and DSP configurations.

```

double * const AAX_RESTRICT deZipper = dzCoefsP->mDeZip [ch ][0];
const double * AAX_RESTRICT coefs = myCoefsP->mBiqCoefsBuf [0];

// Double - precision
for (int i = 0; i < eNumBiquads * eNumCoefs ; ++i)
{
    double dz = deZipper [i];
    dz += zeroCoef * ( coefs [i] - deZipper [i]);
}

```

Listing 20: Example of double-precision smoothing.

In this section we will describe three specific approaches that may be taken to perform optimized real-time smoothing without compromising sound quality.

12.42.8.8.1 Method 1: Clamped single-precision smoothing The simplest approach for optimization of a double-precision smoothing filter is to replace it with modified single-precision smoothing. Unfortunately, we have found that this approach can lead to glitches and instability at higher sample rates when adjusting controls due to transient inaccuracies in the smoothing.

```

double * const AAX_RESTRICT deZipper = dzCoefsP->mDeZip [ch ][0];
const double * AAX_RESTRICT coefs = myCoefsP->mBiqCoefsBuf [0];

// Method 1 - single - precision
for (int i = 0; i < eNumBiquads * eNumCoefs ; ++i)
{
    float dz = deZipper [i];
    dz += zeroCoef * ( coefs [i] - deZipper [i]);

    // If the de -zip step is so small that the coefficient doesn't change then clamp
    // the value to the target to ensure we are using exactly the desired value .
    deZipper [i] = (dz == deZipper [i]) ? coefs [i] : dz;
}

```

Listing 21: Example of clamped single-precision smoothing.

12.42.8.8.2 Method 2: Mixed-precision smoothing To resolve the stability issues at high sample rates, the state may be accumulated at double-precision. This results in mixed-precision operations that are much faster on TI DSPs than full double-precision calculations, though still slower than single-precision.

```
float * const AAX_RESTRICT deZipper = dzCoefsP->mDeZip [ch ][0];
double * const AAX_RESTRICT deZipState = dzCoefsP->mDZState [ch ][0];
const float * AAX_RESTRICT coefs = myCoefsP->mBiqCoefsBuf [0];

// Method 2 - partial double precision
# pragma UNROLL ( CBiquad::eNumCoefs )
for(int i = 0; i < eNumBiquads * eNumCoefs ; i ++){
    {
        double dz = deZipState [i];
        dz += zeroCoef * ((coefs [i]) - ( deZipper [i]));
        deZipState [i] = dz;
        deZipper [i] = float (dz);
    }
}
```

Listing 22: Example of mixed-precision smoothing.

12.42.8.8.3 Method 3: Loop unrolling and double-word memory accesses Further performance gains can be made by unrolling the loop and using double word memory accesses. This code is faster, but is still not as fast as full single-precision.

```
float * const AAX_RESTRICT deZipper = dzCoefsP->mDeZip [ch ][0];
double * const AAX_RESTRICT deZipState = dzCoefsP->mDZState [ch ][0];
const float * AAX_RESTRICT coefs = myCoefsP->mBiqCoefsBuf [0];

// Method 3 - partial double precision - unrolled with double-precision memory accesses
for(int i = 0; i < (eNumBiquads * eNumCoefs); i +=2 )
{
    double dz0 = deZipState [i];
    double dz1 = deZipState [i+1];
    dz0 += zeroCoef * (AAX_LO ( coefs [i]) - AAX_LO ( deZipper [i]));
    dz1 += zeroCoef * ( AAX_HI ( coefs [i]) - AAX_HI ( deZipper [i]));
    deZipState [i] = dz0;
    deZipper [i] = float (dz0);
    deZipState [i+1] = dz1;
    deZipper [i+1] = float (dz1);
}
```

Listing 23: Example of loop unrolling and double-precision memory accesses for smoothing optimization.

12.42.8.8.4 Coefficient smoothing example summary

- Full single-precision smoothing (method 1) is an excellent and simple solution for gain coefficients and other scalar values which are not extremely sensitive to coefficient quantization at small values. This method does not always reach the target value, so clamping should be used to ensure signal integrity.
- Mixed-precision smoothing (method 2) uses slightly more CPU, but gives full double precision accuracy. This approach should generally be used for EQs and other sensitive coefficients.
- Further low-level optimizations are also possible via manual loop unrolling and double-precision memory access (method 3).

12.42.8.9 Refactoring conditionals and branches

Note

For more detailed information on how to reduce or eliminate the use of branches in algorithms, see section 5.2 of the **Hand-Tuning Loops and Control Code on the TMS320C6000** guide provided by TI.

An important technique in refactoring algorithms to enhance loop performance is to reduce or eliminate conditionals and branches in code. The TI compiler focuses a lot of its optimization energy on keeping its pipeline full of inside loops. However, it cannot pipeline a loop if the one of the following is true:

- The loop contains a branch
- The loop contains a function call
- The loop is too long

To demonstrate this, we will again begin with an unoptimized example:

```
for ( int i = 0; i < numSamples ; ++i)
{
    if (! bypass )
    {
        const float filtOutput1 = input [i] * coef0 + state0 * coef1 ;
        const float filtOutput2 = filtOutput1 * coef2 + state1 * coef3 ;
        output [i] = filtOutput2 ;
    }
    else
    {
        output [i] = input [i];
    }
}
```

Listing 24: Another unoptimized filter algorithm.

Though trivial, this example illustrates the problem with conditionals inside of loops. In TI assembly, conditional code usually translates into code branches, which prevents loops from pipelining effectively see [Understanding CGTools-generated ASM files](#). Let's refactor the loop in our example to reduce the size of its conditional branch:

```
for (int i = 0; i < numSamples ; ++i)
{
    const float filtOutput1 = input [i] * coef0 + state0 * coef1 ;
    const float filtOutput2 = filtOutput1 * coef2 + state1 * coef3 ;
    output [i] = filtOutput2 ;

    if ( bypass )
    {
        output [i] = input [i];
    }
}
```

Listing 25: Filter algorithm with a refactored conditional branch.

At first, it may seem wasteful to perform the filter calculation if `bypass` will simply throw away the result. In reality, however, the opposite is true: as a real-time algorithm, this code is constrained by its maximum, worst-case cycle count. It is important to understand this point: essentially, the cycle count of the plug-in is always its worst-case performance.

By reducing the algorithm's maximum cycle count we are therefore reducing waste, even though we are increasing the plug-in's cycle count when it is bypassed. In fact, the ideal scenario for most algorithms is to use only one code path (and, consequentially, a single deterministic cycle count) despite the fact that this can result in worse performance for some specific states. To state this fundamental principle in a different way:

The performance of specific states in an AAX DSP algorithm is not relevant if there is another possible state with worse performance.

Going back to our optimized example, you may also notice that the conditional still exists. Doesn't this create a branch in the assembly code as well and prevent pipelining?

In the case of very brief conditionals such as this, the answer is usually no. On TI processors, most instructions can be executed conditionally, depending on the value of a control register. Thus, the single assignment (`output = input`) inside this conditional will reduce to a few conditional instructions without having to execute a branch. As a result, the TI compiler will be able to efficiently pipeline this loop.

That said, it is occasionally necessary to eliminate conditionals entirely. One effective solution for these situations is to execute the branched logic algorithmically rather than conditionally. To demonstrate this approach, here is our filter example again, this time with the conditional completely eliminated from the loop:

```
for (int i = 0; i < numSamples ; ++i)
{
    const float filtOutput1 = input [i] * coef0 + state0 * coef1 ;
```

```

    const float filtOutput2 = filtOutput1 * coef2 + state1 * coef3 ;
    output [i] = (! bypass ) * filtOutput2 + bypass * input [i];
}

```

Listing 26: Filter algorithm with branching logic executed algorithmically.

This code is shorter and completely eliminates the conditional from inside the loop body. However, there is an associated cost in readability, in that it is not initially obvious how exactly `bypass` affects the output. This is of course a tradeoff that you will need to consider on a case-by-case basis. In general, we encourage you to consider this technique only when you have verified in the assembly code that simply reducing the size of the conditional is not enough to achieve effective instruction pipelining.

Another useful technique for optimizing loops is to use `pragma MUST_ITERATE` and `pragma PROBABLY_ITERATE` (see more about these pragmas in [Loop controls](#)), which help the compiler guess the number of iterations for the loop. It is extremely useful when you know the exact number of the iterations, and this number never changes during plug-in processing. For example, this is applicable for the loops which iterate through the audio samples in the input and output buffers. The number of input samples is always constant for an AAX DSP plug-in algorithm; the buffer length must be described with the option [AAX_eProperty_DSP_AudioBufferLength](#) for each DSP component in the plug-in's description.

The following code example shows an algorithm processing function template. For convenience, this function template takes the audio buffer length as a template parameter:

```

template<int kAudioWindowSize>
void AAX_CALLBACK
Example_AlgorithmProcessFunction( SExampleAlg_Context * const inInstancesBegin [], const void *
    inInstancesEnd)
{
    for (SExampleAlg_Context * const * walk = inInstancesBegin; walk != inInstancesEnd; ++walk)
    {
        SExampleAlg_Context* const AAX_RESTRICT contextP = *walk;
        const float * const AAX_RESTRICT inputP = contextP->mInputPP;
        float * const AAX_RESTRICT outputP = contextP->mOutputPP;

        #pragma MUST_ITERATE( kAudioWindowSize, kAudioWindowSize, kAudioWindowSize )
        for (int32_t i = 0; i < kAudioWindowSize; ++i)
        {
            outputP[i] = inputP[i];
        }
    }
}

```

Listing 27: Optimizing loop using `pragma MUST_ITERATE`.

Note that the audio buffer length property takes a [AAX_EAudioBufferLengthDSP](#) value. The values of this enum are set to the power-of-two for each buffer length, so in this case the `kAudioWindowSize` value would be set to match `2 << AAX_eProperty_DSP_AudioBufferLength` when compiling this algorithm callback into the TI DLL.

The same optimization can be used for the loops that iterate through input/output channels, as demonstrated by the `DemoDist` example plug-in.

12.42.8.10 Case study: pipeline refactoring in Avid's EQ3 and Dyn3 plug-ins

While optimizing the "stock" Pro Tools equalization and dynamics processors we came across many real-world optimization scenarios that will be applicable to a broad variety of plug-ins. In this section we will consider specific techniques that we used to enable software pipelining of these algorithms by the TI compiler, including an in-depth look at the pseudo-speculative execution approach used in our Dyn3 plug-in's polynomial gain calculation loop.

12.42.8.10.1 Move individual processing operations into separate loops Oftentimes a sample-by-sample iterative loop that is not software pipelining can be broken up into individual loops that incrementally apply changes to the audio buffer. These smaller loops have a much better chance of being successfully pipelined by the compiler. In EQ3, moving our biquad audio processing stages to dedicated loops that do not include coefficient smoothing or other tasks resulted in large performance gains.

12.42.8.10.2 Avoid pipeline dependencies The goal of the above optimization is to allow the compiler to successfully pipeline each iterative loop. However, even a pipelined loop may be optimized further. One of the best ways of optimizing loops is to keep the processor busy while pipeline dependencies are cleared.

For example, in EQ3 we found that it was better to perform the plug-in's input and output meter calculations in the same loop rather than separating them out into individual loops. This is because each meter calculation has a dependency on its previous value, which puts a dependency in the pipeline. Doing both at the same time gives the process more to do while waiting for the next value. In Dyn3 we had similar results merging table lookup, attack, and release loops into a single iterative loop. As long as the loop is still successfully pipelined by the compiler, these "larger" loops tended to have much better performance due to the reduction in blocking dependencies.

12.42.8.10.3 Detailed example of loop optimization in Dyn3 At this point it will be helpful to go into greater detail about our optimizations for Dyn3's polynomial gain calculation loop, because the increase in performance was quite large and is fairly representative of other algorithms. The unoptimized code took 43 cycles to execute one iteration of the loop. After rearranging the code it now takes 6 cycles. The basic problem was numerous pipeline dependencies: the *Loop Carried Dependency Bound* was 42 cycles, yet the *Partitioned Resource Bound* was 4 cycles. In other words, if all of these dependencies were removed the loop could potentially execute in 4 cycles.

```

2760 ;* SOFTWARE PIPELINE INFORMATION
2761 ;*
2762 ;* Loop source line : 199
2763 ;* Loop opening brace source line : 200
2764 ;* Loop closing brace source line : 213
2765 ;* Known Minimum Trip Count : 4
2768 ;* Loop Carried Dependency Bound (^) : 42
2769 ;* Unpartitioned Resource Bound : 4
2770 ;* Partitioned Resource Bound (*) : 4
2785 ;*
2786 ;* Searching for software pipeline schedule at ...
2787 ;* ii = 42 Did not find schedule
2788 ;* ii = 43 Schedule found with 1 iterations in parallel
2789 ;* Done

for (int i =0; i< kAudioWindowSize ; i++) // cSmoothingBlockSize
{
    const float * smoothCoeffs = stateP -&gt; mSmoothedPoly ;

    float logEnv = logEnvArray [i]; // logEnvArray [ fIdx +i];
    logEnv -= smoothThrLow ;

    if( logEnv >= 0.0 f) // In the knee
        smoothCoeffs += eCpdPolyOrder ;
    if( logEnv >= 0.0 f) // In the knee
        logEnv -= smoothThrLowDelta ;
    if( logEnv >= 0.0 f) // In the linear GR stage
        smoothCoeffs += eCpdPolyOrder ;

    const float filteredLogEnv = smoothCoeffs [ eCpdPolyCoeffsC ] +
        logEnv *( smoothCoeffs [ eCpdPolyCoeffsB ] +
            smoothCoeffs [ eCpdPolyCoeffsA ]* logEnv );
    filtLogEnvArray [i] = filteredLogEnv + smoothedMakeupGain ;
}

```

Listing 28: Dyn3's unoptimized polynomial gain calculation loop and asm listing.

- `logEnv -= smoothThrLow` *depends on the result of logEnvArray[i]*
- `if(logEnv >= 0.0f)` *depends on the result of logEnv -= smoothThrLow*
- `logEnv -= smoothThrLowDelta` *depends on the result of logEnv -= smoothThrLow*
- `Thrid if(logEnv >= 0.0f)` *depends on the result of logEnv -= smoothThrLowDelta*
- `Second smoothCoeffs += eCpdPolyOrder` *depends on the result of the first smoothCoeffs += eCpdPolyOrder*
- `logEnv*smoothCoeffs[eCpdPolyCoeffsB]` *depends on the result of logEnv -= smoothThrLowDelta*
- `smoothCoeffs[eCpdPolyCoeffs], etc.` *depend on the result of the second smoothCoeffs += eCpdPolyOrder*

- `filteredLogEnv+smoothedMakeupGain` *depends on the result of* `filteredLogEnv = smoothCoeffs[eCpdPolyCoeffsC]`
- `filtLogEnvArray[i]` *depends on the result of* `filteredLogEnv + smoothedMakeupGain`

And I don't think that even covers every case, but you get the idea. The bottom line is there is no way this loop can pipeline well. In contrast, here is the optimized code and listing file output once these dependencies have been removed:

```

2476 ;* Loop opening brace source line : 167
2477 ;* Loop closing brace source line : 179
2446 ;* Known Minimum Trip Count : 4
2482 ;* Loop Carried Dependency Bound (^) : 1
2483 ;* Unpartitioned Resource Bound : 4
2484 ;* Partitioned Resource Bound (*) : 4
2512 ;* ii = 6 Schedule found with 5 iterations in parallel

for (int i =0; i< cProcessingBlockSize ; i++)
{
    float logEnv = logEnvArray [i];
    float logEnvThrHi = logEnv - smoothThrHigh ;
    const float gainSlope = smoothThrSlope +
        logEnv * smoothSlope ;
    const float gainKnee = smoothKneeC +
        logEnvThrHi *( smoothKneeB +
            smoothKneeA * logEnvThrHi );

    const bool bKnee = ( logEnv > smoothThrLow );
    const bool bSlope = ( logEnv > smoothThrHigh );

    float filteredLogEnv = bKnee ? gainKnee : 0.0f;
    filteredLogEnv = bSlope ? gainSlope : filteredLogEnv ;
    filtLogEnvArray [i] = filteredLogEnv ;
}

```

Listing 29: Dyn3's optimized polynomial gain calculation loop and asm listing

In this case `gainSlope` is only dependent on the loading of `logEnv`, so that can begin almost immediately. `GainKnee` must wait for `logEnvThrHi`, but `gainSlope` can be calculated during that time. `bKnee` and `bSlope` are also only dependent on `logEnv`, and start right away. The main dependency is `filteredLogEnv` which is dependent on `bKnee` and `gainKnee` and then `bSlope` and `gainSlope`. Anyhow, this is far fewer dependencies. Here is another version which runs in exactly the same number of cycles. (In fact, under the hood it may be creating the same asm code; we have not compared instruction-by-instruction.)

```

for (int i =0; i< kAudioWindowSize ; i++)
{
    float logEnv = logEnvArray [i];
    float logEnvThrHi = logEnv - smoothThrHigh ;

    const bool bKnee = ( logEnv > thrLow );
    const bool bSlope = ( logEnv > thrHigh );

    float filteredLogEnv = bKnee ?
        kneeC + logEnvThrHi *( kneeB + kneeA * logEnvThrHi ) :
        0.0 f;
    filteredLogEnv = bSlope ?
        thrSlope + logEnv * slope :
        filteredLogEnv ;
    filtLogEnvArray [i] = filteredLogEnv ;
}

```

Listing 30: An alternative optimization for Dyn3's polynomial gain calculation loop.

12.42.8.10.4 But what about Native? You might expect this altered code to execute well on a TI DSP but poorly on x86. However, keep in mind that a large degree of speculative execution is used on Intel's processors. This means that pipeline dependencies due to conditionals can be broken because multiple paths are executed. In these cases, only one of the results is used and the others are thrown away. In other words, if you saw pseudo code showing the literal execution of the unoptimized code above on Intel then it would probably look a lot like the optimized code. The lesson? For TI it is important to rearrange your code so that essentially it implements speculative execution as much as possible, and if applied correctly this optimization should not negatively impact your plug-in's native performance.

12.42.8.11 Case study: Additional optimization lessons from EQ3 and Dyn3

The pipeline optimization example above is just one example, and the following techniques also helped us achieve many-fold increases in performance. Note that many of these techniques are discussed in greater detail in the sections above.

12.42.8.11.1 Watch the assembly listing In the process of optimizing these plug-ins we found their asm listing files very helpful, especially the *Loop Carried Dependency Bound* and the *Partitioned Resource Bound* information. The listing file shows how many cycles the code is taking to execute, and we could make an estimate of how far away we were from the optimal implementation by seeing how well the pipeline is being utilized.

12.42.8.11.2 Divide processing tasks over multiple calls In the old RTAS version of EQ3 the coefficients were updated (smoothed) every 8 samples. Initially, this was changed to every 4 samples in the AAX version in order to easily work with 4-sample blocks on HDX. However, we were able to achieve better results by adding "ping pong" logic that alternates between smoothing the first and second half of the coefficients on each pass. To make this work in our odd-banded EQ we had to pad the smoothing coefficients by one biquad's worth to make an even number of biquads, but regardless of this inefficiency we still achieved performance gains.

12.42.8.11.3 Eliminate branches that block pipelining Eliminating large conditional branches is critical to optimal performance on TI. This can be an especially tempting pitfall for developers who are used to coding only for x86 processors.

Consider the "ping pong" optimization described above. This logic does not break pipelining because the conditional logic that checks the state of the flag does not result in a large branch; once the ping pong value is set, the exact same logic operates in every processing callback. If instead we used an if statement to determine which "side" should execute, this would prevent pipelining optimizations and would seriously impact performance.

12.42.8.11.4 Remove double-precision operations where they are not required Here is some coefficient smoothing code from our pre-optimization EQ3 algorithm. This code was embedded in the inner biquad processing loop:

```
# pragma UNROLL ( CBiquad::eNumCoefs )
for (int k = 0; k < CBiquad::eNumCoefs; ++k)
{
    double &dz = deZipper[k];
    AAX::DeDenormal (dz);
    step[k] = zeroCoef * ( coefs[k] - dz);
}

# pragma UNROLL ( CBiquad::eNumCoefs )
for(int k = 0; k < CBiquad::eNumCoefs; ++k)
{
    double nml_dz = deZipper[k]; // read state
    nml_dz += step[k];
    biquadCoefs[k] = static_cast< float > ( nml_dz );
    deZipper[k] = nml_dz ; // write state
}
```

Listing 31: Unoptimized coefficient smoothing in EQ3

To optimize this code, we converted the logic to use single-precision de-zipper values. However, this resulted in a sonic difference due to the fact that the smoothed coefficients would not necessarily ramp all the way to the correct target value. To solve that we added a conditional "clamp" that halts the smoothing once there is no difference between the 32-bit smoothed value and the target value. On examination of the assembler output, we found that this conditional pipelines very well.

```
# pragma UNROLL ( CBiquad::eNumCoefs )
for(int i = 0; i < (cMaxNumBiquadsWithPad / 2) * CBiquad::eNumCoefs; ++i)
{
    float dz = deZipper[i];
    dz += zeroCoef * ( coefs[i] - deZipper[i]);
    deZipper[i] = (dz == deZipper[i]) ? coefs[i] : dz; // clamp
}
```

Listing 32: Optimized coefficient smoothing in EQ3

12.42.8.11.5 Make coefficients contiguous We were able to achieve significant performance gains in iterative loops like the smoothing code shown above by ensuring that all of the coefficients that would be accessed by the loop are contiguous in memory. In addition, note that in the optimized code there is only one loop, which iterates `NumBiquads*NumCoefs` times. This optimization is possible due to the fact that each filter's coefficients are contiguous in the `coefs` array.

12.42.8.11.6 Use AAX_RESTRICT wherever applicable We have found that the `restrict` keyword is vital for optimal performance on TI DSPs. For example, the parameter smoothing logic in our Dyn3 plug-in was reduced from 18 cycles to 3 cycles per loop iteration simply by the addition of this keyword to the applicable pointer variables.

For more information about the `restrict` keyword, see [restrict](#).

12.42.8.11.7 Be aware of shell overhead In the TI Shell there is code that loops through every buffered coefficient FIFO before every sample buffer in order to swap the algorithm's context field pointers to a new set of coefficients if one is available. This uses a nominal number of cycles per buffered port, which can add up very quickly in small plug-ins.

For example, before our optimizations EQ3 used eight individual buffered coefficient blocks. On investigation, we found that the shell overhead from managing these buffers added up to be roughly equivalent to the algorithm's total processing cycles! To work around this we merged the 8 coefficient blocks into one large block. The trade-off of this optimization is that more work must be done on the host to re-generate and copy the whole coefficient state every time any parameter changes, so this is an optimization that should be applied only when appropriate for the individual plug-in. For example, before our optimizations EQ3 used eight individual buffered coefficient blocks. On investigation, we found that the shell overhead from managing these buffers added up to be roughly equivalent to the algorithm's total processing cycles! To work around this we merged the 8 coefficient blocks into one large block. The trade-off of this optimization is that more work must be done on the host to re-generate and copy the whole coefficient state every time any parameter changes, so this is an optimization that should be applied only when appropriate for the individual plug-in.

12.42.8.11.8 Watch for opportunities to merge or eliminate operations Keep an eye out for unnecessary processing stages performed by your algorithm. Gain stages, phase toggles, and "dummy" coefficients are particularly good candidates for this kind of optimization. For example:

- In our EQ3 plug-in, we found that we could achieve significant performance improvement by merging the plug-in's input and output gain stages with the overall gain of the first and last biquads. As a side benefit, this reduced the total quantization noise in the algorithm.
- In our Dyn3 plug-in, we found that we were applying smoothing logic to filter coefficients that would always be zero.
- When we looked more closely at Dyn3 we found that we were also computing and discarding sidechain filter information for the LFE, which is not part of the sidechain

12.42.8.11.9 Read the TI documentation There are many helpful optimization resources available from Texas Instruments. Out of all of the TI optimization documents we encountered, we found the *Hand-Tuning Loops and Control Code on the TMS320C6000* guide to be the most helpful and complete.

12.42.8.12 Optimization on the HDX platform

12.42.8.12.1 Interrupt latency Besides the large latency due to context switching (lots of data file registers to store) and the pipeline (many stages), interrupts can be disabled around pipelined loops, which cannot be interrupted. This can be controlled with the `-mi=X` compiler option, which will disallow unsafe pipelining for loops that are longer than X cycles. See TI's documentation (SPRU187O Section 2.12) for more details and references regarding this behavior.

12.42.8.12.2 External memory access A loop which performs many reads and writes may require access to external memory. In this scenario, the loop may take 10's or even 100's of times longer to execute than the compiler expects it to!

There are two options for dealing with this:

1. Search and destroy these loops individually
 - Move all the data used by the loop to internal RAM.
 - Use HDX's DMA facilities for external memory accesses.
 - `#pragma FUNC_INTERRUPT_THRESHOLD` can be used to disable pipelining on a case by case basis.
2. For modules that are known to have these loops but are not worth hand optimizing, then turn off pipelined loop optimization altogether. (`-mu` aka `-disable_software_pipelining`).

Note

This is only a problem in the C67(0-2)x ISAx used on the HDX platform. In The C64xx and C674x ISA, there is an SPLOOP command which can buffer the branches within pipelined loops to allow them to be interruptable.

12.42.8.13 Code Composer Studio optimization tools

12.42.8.13.1 Compiler Consultant The Compiler Consultant tool can be used to suggest additional optimizations.

To enable the Compiler Consultant in Code Composer Studio, do the following:

1. Set an optimization level of `-o2` or `-o3` (Found in CCSv4 under Build Options > Compiler > Basic)
2. Set the `-consultant: Generate Compiler Consultant Advise` switch (Found in CCSv4 under Build Options > Compiler > Feedback)

12.42.8.13.2 Optimization information file Optimization information files can be generated in Code Composer Studio by selecting the option Build Options > Compiler > Feedback > Opt Info File. Optimization information files have an .nfo extension and are placed into the project's intermediate build products directory. In general, these files list function call-graph information and describe whether or not individual functions can be inlined.

12.42.9 Error Codes

The following appendices document error codes that are specific to plug-in hosting in Pro Tools HDX and other AAX platforms based on the TI DSP environment.

12.42.9.1 -138xx: DHM Core DSP errors

These errors relate to routing and assignment problems on Pro Tools HDX hardware. Plug-ins should never be able to trigger these error codes, which indicate low-level problems in the system.

Table 1: DHM Core DSP error codes	
Value	Definition
-13801	ePSError_CTIDSP_WrongSampleRate
-13802	ePSError_CTIDSP_NoFreeStreams
-13803	ePSError_CTIDSP_StreamCreationTimeout
-13804	ePSError_CTIDSP_StreamDestruction
-13805	ePSError_CTIDSP_InactiveStream
-13806	ePSError_CTIDSP_StreamCorrupted
-13807	ePSError_CTIDSP_QueueFull
-13808	ePSError_CTIDSP_NullPointer
-13809	ePSError_CTIDSP_WrongStreamID
-13810	ePSError_CTIDSP_ImageError
-13811	ePSError_CTIDSP_ResetError
-13812	ePSError_CTIDSP_ImageVerify
-13813	ePSError_CTIDSP_DSPAlreadyInBootOrReset
-13814	ePSError_CTIDSP_TriggerInterrupt
-13815	ePSError_CTIDSP_BufferSizeNotAligned
-13816	ePSError_CTIDSP_TimeoutWaitingForHPIC
-13817	ePSError_CTIDSP_SetUHPIError
-13818	ePSError_CTIDSP_UHPINotReady

12.42.9.2 -140xx: AAX Host errors

These errors relate to logic failures in the AAX host software. These errors can be due to plug-in bugs or system configuration problems.

Table 2: AAX Host Software error codes	
Value	Definition
-14001	kAAXH_Result_Warning
-14003	kAAXH_Result_UnsupportedPlatform
-14004	kAAXH_Result_EffectNotRegistered
-14005	kAAXH_Result_IncompleteInstantiationRequest
-14006	kAAXH_Result_NoShellMgrLoaded
-14007	kAAXH_Result_UnknownExceptionLoadingTIPlugIn
-14008	kAAXH_Result_EffectComponentsMissing
-14009	kAAXH_Result_BadLegacyPlugInIDIndex
-14010	kAAXH_Result_EffectFactoryInitiatedTooManyTimes
-14011	kAAXH_Result_InstanceNotFoundWhenDeinstantiating
-14012	kAAXH_Result_FailedToRegisterEffectPackage
-14013	kAAXH_Result_PlugInSignatureNotValid
-14014	kAAXH_Result_ExceptionDuringInstantiation
-14015	kAAXH_Result_ShuffleCancelled
-14016	kAAXH_Result_NoPacketTargetRegistered
-14017	kAAXH_Result_ExceptionReconnectingAfterShuffle
-14018	kAAXH_Result_EffectModuleCreationFailed
-14019	kAAXH_Result_AccessingUninitializedComponent
-14020	kAAXH_Result_TIComponentInstantiationPostponed

-14021	kAAXH_Result_FailedToRegisterEffectPackageNotAuthorized
-14022	kAAXH_Result_FailedToRegisterEffectPackageWrongArchitecture
-14023	kAAXH_Result_PluginBuiltAgainstIncompatibleSDKVersion
-14023	kAAXH_Result_PluginBuiltAgainstIncompatibleSDKVersion
-14100*	kAAXH_Result_InvalidArgumentValue
-14101*	kAAXH_Result_NameNotFoundInPageTable

*Overlaps with -141xx: [TI System errors](#) definitions

12.42.9.3 -141xx: TI System errors

These errors relate to logic failures in the TI management software and generally indicate a failure in the HDX system services such as buffered message queues, context management, and callback timing.

Table 3: TI system error codes	
Value	Definition
-14101	eTISysErrorNotImpl
-14102	eTISysErrorMemory
-14103	eTISysErrorParam
-14104	eTISysErrorNull
-14105	eTISysErrorCommunication
-14106	eTISysErrorIllegalAccess
-14107	eTISysErrorDirectAccessOfFifoBlocksUnsupported
-14108	eTISysErrorPortIdOutOfBounds
-14109	eTISysErrorPortTypeDoesNotSupportDirectAccess
-14110	eTISysErrorFIFOFull
-14111	eTISysErrorRPCTimeOutOnDSP
-14112	eTISysErrorShellMgrChip_SegsDontMatchAddrs
-14113	eTISysErrorOnChipRPCNotRegistered
-14114	eTISysErrorUnexpectedBufferLength
-14115	eTISysErrorUnexpectedEntryPointName
-14116	eTISysErrorPortIDTooLargeForContextBlock
-14117	eTISysErrorMixerDelayNotSupportedForPlugIns
-14118	eTISysErrorShellFailedToStartUp
-14119	eTISysErrorUnexpectedCondition
-14120	eTISysErrorShellNotRunningWhenExpected
-14121	eTISysErrorFailedToCreateNewPIInstance
-14122	eTISysErrorUnknownPIInstance
-14123	eTISysErrorTooManyInstancesForSingleBufferProcessing
-14124	eTISysErrorNoDSPs
-14125	eTISysBadDSPID
-14126	eTISysBadPIContextWriteBlockSize
-14128	eTISysInstanceInitFailed
-14129	eTISysSameModuleLoadedTwiceOnSameChip
-14130	eTISysCouldNotOpenPlugInModule
-14130	eTISysCouldNotOpenPlugInModule
-14131	eTISysPlugInModuleMissingDependencies
-14132	eTISysPlugInModuleLoadableSegmentCountMismatch
-14133	eTISysPlugInModuleLoadFailure
-14134	eTISysOutOfOnChipDebuggingSpace
-14135	eTISysMissingAlgEntryPoint
-14136	eTISysInvalidRunningStatus
-14137	eTISysExceptionRunningInstantiation
-14138	eTISysTIShellBinaryNotFound

-14139	eTISysTimeoutWaitingForTIShell
-14140	eTISysSwapScriptTimeout
-14141	eTISysTIDSPModuleNotFound
-14142	eTISysTIDSPReadError

12.42.9.4 -142xx: DIDL errors

These errors all relate to the dynamic library loading system that manages ELF DLL binaries on Pro Tools HDX hardware. For example, a `eDIDL_FileNotFound` error will be raised if the ELF DLL name specified by an Effect's Describe code does not match any DLL that is present in the plug-in's bundle.

Table 4: DIDL error codes	
Value	Definition
-14201	eDIDL_FileNotFound
-14202	eDIDL_FileNotOpen
-14203	eDIDL_FileAlreadyOpen
-14204	eDIDL_InvalidElfFile
-14205	eDIDL_ImageNotFound
-14206	eDIDL_SymbolNotFound
-14207	eDIDL_DependencyNotLoaded
-14208	eDIDL_BadAlignment
-14209	eDIDL_NotImplemented

12.42.9.5 -144xx: HDX hardware errors

These errors relate to failures on the HDX hardware itself. Plug-ins should never be able to trigger these error codes, which indicate low-level problems in the system.

Table 5: HDX hardware error codes	
Value	Definition
-14401	eBerlinImageError
-14402	eBerlinImageWriteError
-14403	eBerlinInvalidArgs
-14404	eBerlinCantGetTMSChannel
-14405	eBerlinChunkWriteError
-14406	eBerlinChunkReadError
-14407	eBerlinInvalidReqID
-14408	eBerlinDSPInResetError
-14409	eBerlinDSPTimeOut
-14410	eBerlinIncorrectTdmCableWiring
-14411	eBerlinInvalidClock

12.42.9.6 -145xx: DHM isochronous audio engine errors

These errors relate to failures within the HDX audio engine software. Plug-ins should never be able to trigger these error codes, which indicate low-level problems in the system.

Table 6: DHM isochronous audio engine error codes

Value	Definition
-14500	eDsiIsochEngineGenericError
-14501	eDsiIsochEngineWrongChannelNumber
-14502	eDsiIsochEngineTxRingFull
-14503	eDsiIsochEngineRxRingNotReady
-14504	eDsiIsochEngineWrongNumberOfSamplesRequest
-14505	eDsiIsochEngineUnrecognizedSampleRate
-14506	eDsiIsochEngineUnsupportedSampleSizeBytes
-14507	eDsiIsochEngineUnsupportedNumberOfChannels
-14508	eDsiIsochEngineUnsupportedSampleRate
-14509	eDsiIsochEngineDMAAlreadyEnabled
-14510	eDsiIsochEngineDMAAlreadyDisabled
-14511	eDsiIsochEngineInterruptHandlerAlreadyInstalled
-14512	eDsiIsochEngineBadCardRecord
-14513	eDsiIsochEngineCantSetValueDuringStreaming
-14514	eDsiIsochEngineStreamingAlreadyStarted
-14515	eDsiIsochEngineStreamingAlreadyStopped
-14516	eDsiIsochEngineStreamingCantBeStarted
-14517	eDsiIsochEngineUnsupportedSamplesPerInterrupt
-14518	eDsiIsochEngineCantSetSamplesPerInterrupt
-14519	eDsiIsochEngineInterruptLoopAlreadyExists
-14520	eDsiIsochEngineGlobalDMADisabled
-14521	eDsiIsochEngineActiveInterruptMaskAlreadyEnabled
-14522	eDsiIsochEngineSDIOErrors

12.42.9.7 -30xxx: Dynamically-generated error codes

Errors in the -30xxx range are dynamically generated codes, and thus the same failure point could generate a different error code depending on the order in which errors occurred. These kinds of error codes are used heavily by the TI Shell Manager, the host component that interacts with the on-DSP shell environment.

If one of these error codes is being generated by the TI Shell Manager (the most common case) then you should be able to get more information about the failure by enabling the following [DigiTrace](#) logging facility:

```
DTF_TISHELLMGR=file@DTP_NORMAL
```

or, within the DSH tool:

```
enable_trace_facility [DTF_TISHELLMGR, DTP_NORMAL]
```

This should result in a log with more information such as the name of the failing plug-in, the dynamically generated error code, and a string description of its meaning. Depending on the failure case, the DAE dish command `getlastdsploadererror` can also sometimes be used to retrieve the description string for a dynamically-generated error if it was the last error generated during the DSP loading operation.

Collaboration diagram for HDX DSP Guide:



12.43 Page Table Guide

How to map a plug-in's parameters to control surfaces.

12.43.1 Contents

- [Introduction](#)
- [Avid Control Surfaces](#)
- [Plug-In Page Table Guidelines](#)
- [Avid Center Section Page Tables](#)
- [EUCON Page Tables](#)
- [Implementing Page Tables](#)
- [Appendix A. Get Parameter Value Info](#)

12.43.2 Introduction

12.43.2.1 Control Surfaces Overview

A tactile, external hardware control surface can be used to control different aspects of an application such as Pro Tools or Media Composer. Users prefer purpose-built control surfaces for DAW manipulation due to the control surface's superior accessibility, tactile feel, ergonomics, and user feedback.

Avid provides several different control surface products designed to accommodate a wide variety of user needs and workflows. Most Avid control surfaces implement EUCON (Extended User Control), a high-speed open control protocol featuring high-resolution, responsive control over almost all software functions. Some other control surfaces, such as the C|24, implement a dedicated control protocol for direct integration with Pro Tools. Finally, Pro Tools includes support for Mackie's HUI protocol and can interface with third-party control surfaces that implement this protocol.

12.43.2.2 Page Tables Overview

When a control surface is attached to a DAW system running AAX plug-ins the surface can be used to manipulate plug-ins' parameters. Plug-ins define the mapping between their parameters and control surface encoders using *page tables*.

Abstractly, a page table is a static mapping of a plug-in's controls to the interface of the control surface. Since a plug-in may have many more controls than the control surface can accommodate at a given time, the controls may be split across several "pages" that the user can freely switch between.

More concretely, a set of page tables is simply a set of single dimensional arrays. Each slot of the array corresponds to a particular rotary encoder or push-button of the control surface. By inserting control indices into the elements of the array, a plug-in's controls are mapped to particular tangible controls of the CS. Page tables are stored as XML data created by the [Page Table Editor](#) application available as part of the [AAX SDK Toolkit](#) on the [My Toolkits and Downloads](#) page at avid.com. The XML is referenced by the plug-in as a resource using a call to `AAX_IEffectDescriptor::AddResourceInfo()`.

The following sections describe the various interfaces that the supported control surfaces provide for modifying plug-in parameters. Later, the specifics of implementation of page tables is described.

12.43.3 Avid Control Surfaces

12.43.3.1 EUCON

12.43.3.1.1 Pro Tools | Control app and Pro Tools | Dock The free Pro Tools | Control app provides a EUCON-enabled control surface for iPad. Combining Pro Tools | Control with Pro Tools | S3, Pro Tools | Dock, or Artist Mix hardware adds additional touch workflows and custom control. Pro Tools | Control app display controlling an EQ plug-in instance

Pro Tools | Dock connects with an iPad and the free Pro Tools | Control iOS app, providing intelligent control of audio and video projects. The app offers a host of touch controls and visual feedback. The Dock provides eight push-top, touch-sensitive Soft Knobs that interact with whatever knobset has been chosen in the Pro Tools | Control app. Select an EQ, plug-in, send, pan, or other item, and all parameters instantly map to the knobs for tweaking. Pro Tools | Dock with iPad and Pro Tools | Control app

When laying out plug-in parameters on its display, Pro Tools | Control uses the same page table layout as [Artist Control](#)

12.43.3.1.2 Pro Tools | S6 Pro Tools | S6 is a modular control surface solution for the most demanding audio mixing and production environments. Built on the same proven technology that is core to the industry-leading ICON and System 5 product families, S6 enables mixers to quickly turn around complex projects while swiftly handling last-minute changes. With its unparalleled ability to simultaneously control multiple Pro Tools and other EUCON-enabled DAWs over simple Ethernet, S6 also speeds workflows and enables network collaboration on a single integrated platform. Pro Tools | S6

The main touch screen display on S6 can be configured to show graphs representing a plug-in's frequency or dynamics response curves. To support this feature, a plug-in must implement [AAX_IEffectParameters::GetCurveData\(\)](#) for the applicable [AAX_ECurveType](#) selector.

12.43.3.1.3 Pro Tools | S3 Pro Tools | S3 is a compact, EUCON-enabled, ergonomic desktop control surface that offers a streamlined yet versatile mixing solution for the modern sound engineer. Like S6, S3 delivers intelligent control over every aspect of Pro Tools and other DAWs, but at a more affordable price. While its small form factor makes it ideal for space-confined or on-the-go music and post mixing, it packs enormous power and accelerated mixing efficiency for faster turnarounds, making it the perfect fit everywhere, from project studios to the largest, most demanding facilities. Pro Tools | S3

On the S3 control surface, the top-right eight encoder/OLED pairs may be dedicated to any plug-in instance. In addition, the S3 includes support for dedicated EQ and dynamics plug-ins: a user can select a particular plug-in as his EQ or dynamics processor and use the surface's EQ, Compressor/Limiter, and Expander/Gate selectors to bring up the plug-in in a separate group of eight encoders. This mode uses the plug-in's D-Control Center Section EQ or Dynamics page tables when mapping plug-in controls. In order to be selectable as the system's EQ or dynamics processor a plug-in must meet the criteria for an ICON center-section plug-in.

12.43.3.1.4 Avid Artist Series: Artist Control The Artist Control can run as either a standalone device or connected to additional units to form a larger system for your favored DAW. EUCON Artist Series: Artist Control

The plug-in editing section for the Artist Control consists of 8 touch and velocity sensitive rotary encoders, and 8 touch screen switches. The rotary knobs are physically laid out as two groups of 4, vertically aligned and positioned next to the customizable LED-backlit touch-screen interface, with their corresponding touch screen switches right next to them. The alpha-numeric scribble strip and plug-in editing displays are 8 characters wide. The knobs and switches can be assigned using the [Av81](#) page table.

Mapping of plug-in parameters to the surface's various controllers can also be done through use of the ProControl or D-Control page tables, giving the Artist Control the ability to emulate various plug-in editing sections (D-Control's Center Section EQ/Dynamics Sections and Channel Strip) or as a dedicated plug-in editing section (ProControl). Consequently, if using the ProControl page table, plug-in developers will have access to 16 controls, rotary encoders numbered top to bottom, left to right, as 0 through 7, and touch screen switches numbered top to bottom, left to right, as 8 through 15.

12.43.3.1.5 Avid Artist Series: Artist Mix The Artist Mix can run as either a standalone device or connected to additional units to form a larger system for audio mixing applications.

EUCON Artist Series: Artist Mix

The plug-in editing section for the Artist Mix consists of 8 touch and velocity sensitive rotary encoders, and 8 switches. The rotary knobs are physically laid out horizontally along the top of the control surface as a group of 8, located right below the display screen, with the switches located directly below the encoders (the "ON" buttons). The alpha-numeric scribble strip and plug-in editing display are 8 characters wide. The Artist Mix is mapped using the Av18 page table, with the recommendation that the most important parameter is listed first. Like the Artist Control, mapping of plug-in parameters to the surface's various controllers can be done through use of the ProControl and/or D-Control page tables. This allows the Artist Mix to emulate the various plug-in editing sections that pertain to the D-Control, as well as the ProControl's dedicated plug-in section. Both rotary encoders and switches are numbered right to left, as 0 through 7, and 8 through 15 respectively.

12.43.3.1.6 Avid Pro Series: System 5 The System 5 Avid series consists of digital audio mixing systems that can be configured with hundreds of channels. These systems are used specifically for audio post production and music applications. Channels have a 4 band EQ, dynamics, two filters, and consist of 8 rotary knobs and a touch-sensitive motorized fader.

The plug-in editing section consists of 8 knob/switch pairs. This section can take its mapping from the Av81 page table, with the recommendation that the most important parameter is last (closer to the operator). Like the Artist Series of EUCON controllers, mapping of plug-in parameters can also be accomplished through use of either ProControl or D-Control page tables. When using these page tables, the plug-in editing section is numbered top to bottom, 0 through 7, and 8 through 15, respectively.

The System 5 can also be put into a mode where a single plug-in is mapped across 4 rows of rotary knobs across an entire 8-fader bank. This mode uses the Av48 page table.

12.43.3.1.7 Avid Pro Series: MC Pro The MC Pro is a workstation control surface that is geared towards professional post production. As the professional version of Avid's Artist Series Artist Control, it delivers precise and fast control of your editing applications.

The plug-in editing section consists of 8 pairs of touch-sensitive rotary knobs and switches. The rotary knobs are equipped with LED display rings that are available for control of EQ, Dynamics or any type of control your plug-in may need. As with the Artist Control, plug-in parameters can be mapped with the EUCON Av81 page table or by the ProControl or D-Control page tables described below. The 8 knobs are placed as two vertical groups of 4, laid out on the left half of the control surface, and are numbered top to bottom, left to right.

12.43.3.2 Avid C|24 and ICON

12.43.3.2.1 C|24 C|24 is a hardware control surface allowing great flexibility and power to Pro Tools systems. The alphanumeric scribble strip and plug-in editing displays allow 4 characters each for the plug-in name and a control's name. Three characters are allowed for the control's value.

C|24

The plug-in editing section consists of 24 horizontally aligned rotary data encoders and 24 switches lined up under the encoders. (the image above shows only 12 so that more detail can be displayed.) The C|24 encoders and switches are set up as pairs.

In any given control pair, either the encoder or a mix of encoder + the switch is used depending on how the mapped parameter has been defined in the plug-in:

- A parameter defined as a discrete parameter is automatically assigned to the switch and encoder.
 - The switch will increment the value of the parameter each time it is pressed.
 - If a discrete parameter has only two possible values, the switch's backlight will be illuminated when the parameter has a non-zero value ("On")
- A parameter defined as a continuous will automatically be assigned to the rotary encoder.

Therefore, whether the plug-in has all switches, all encoders, or a mix, there are 24 controls available per page on the C|24.

12.43.3.2.2 ICON: D-Control ES D-Control is a modular hardware control surface that adds high-quality tactile mixing and editing capability to Pro Tools systems. D-Control is a high-end mixing system that includes a center section and one or more fader packs. A fader pack consists of 16 channel strips, displaying track and plug-in information. Further general information about D-Control and how it works is available from the "BuckleyBriefing.pdf" on the developer website.

D-Control has the potential of displaying plug-in controls in four different sections: Channel Strip, Custom Fader, Center Section EQ, and Center Section Dynamics. The Dynamics section actually makes use of two different page table types, while the other three each have one page table type. In all sections, scribble strips are 6-characters wide and do not support the Digidesign Extended character set (i.e. special characters).

- Channel Strip

Plug-ins' controls are displayed in the channel strip section of the D-Control. Each channel strip consists of 6 vertically aligned encoder/switch pairs. This gives a total of 12 controls per page, where the encoders (from top to bottom) are numbers 1-6, and the switches (from top to bottom) are numbered 7-12. Like ProControl, the lower numbered encoder is paired with the lower numbered switch. In the diagram to the right, the button labeled "B-M-P" will control the switch.

There are a couple of special considerations when making D-Control Channel Strip page tables. First, keep in mind that the strips are at the top half of the control surface. Therefore, it will be more convenient for the user if you place the most frequently used controls at the bottom rather than at the top. Second, when a control is present in the switch of an encoder / switch pair, the "Pre" light will be lit as an indication to the user of the switch's presence. However, the name of the control placed on the switch will not appear in the scribble strip, unless the user presses the Sw Info button. The value of the switch will appear when the switch is pressed, but the name does not appear. Therefore, it is not recommended that you place a switch next to an empty encoder. It is recommended that you either place a switch next to an encoder that makes sense (like a Gain encoder next to a Phase switch) or place the switch control in both the switch and encoder so that the user will see the name of the control. These are merely guidelines, and not hard and fast rules.

D-Control: Channel Strip

- Custom Fader

D-Control can be placed in a special mode called Custom Fader to allow more controls of a plug-in to be displayed at once. D-Control takes the currently selected plug-in and displays its controls across 8 of the 16 channel strips such that the plug-in's first control is in the lower left control, its eighth is in the lower right. If there is more than one page of controls, D-Control displays additional pages, up to six, in the rows above the first. If the user has more than one fader pack, the Custom Faders can spread to more sets of faders.

D-Control: Custom Fader

D-Control: Custom Fader Page Layout

To support this mode, a plug-in must include a page table with 16 controls (8 encoder/switch pairs) per page. See the diagram to see how the layout of several Custom Fader page tables works.

Since the one page layout is similar to the ProControl layout (16 controls per page with 8 encoders and 8 switches), D-Control will use your plug-in's ProControl page tables as a default. If you are not happy with how your plug-ins controls appear in this manner, you can override this by creating a Custom Fader page table.

- Center Section EQ and Dynamics Sections

D-Control's center section contains two dedicated areas for EQ and Dynamics plug-ins. Only plug-ins falling into these categories (EQ, Compressors, Limiters, Expanders, and Gates) should implement page tables for these sections. If your plug-in does not fall into these categories, you can skip these page table types. The remainder of the descriptions of these types can be found below in the Center Section Page Tables section.

12.43.3.2.3 ICON: D-Command ES D-Command ES is a more compact yet expandable console than the D-↔ Control. Coming standard with 8 physical faders/channel strips (two encoders per strip), it is expandable to 40 faders/channels.

The plug-in editing section for the D-Command follows closely to that of the D-Control, with two encoders per channel strip. It provides dedicated Dynamics and EQ sections for plug-ins that support Dynamics and EQ plug-in mapping, as well as the ability to display plug-ins in a Custom Fader section as described in the D-Control section. All sections provide scribble strips that 6 alpha numeric characters wide.

12.43.3.3 VENUE

VENUE is a revolutionary line of digital live sound systems that deliver amazing sound quality, reliability, flexibility, and ease-of-use. These live sound systems come equipped with plug-in editing sections, and work together to deliver studio-grade sound and powerful performance. For more information about using AAX plug-ins with VENUE systems see the [VENUE Guide](#).

12.43.3.3.1 VENUE | S6L VENUE | S6L is a modular system designed to take on the world's most demanding tours and events with ease. Offering unprecedented processing capabilities - with over 300 processing channels - S6L delivers unrelenting performance and reliability through its advanced engine design and backs it up with modern touchscreen workflows and scalability to meet any challenge. VENUE | S6L console

S6L uses the same modular components as [Pro Tools | S6](#), but uses a different set of encoder layouts on these components in order to best support mixing workflows in a live sound setting. When displaying plug-in parameters, S6L maps the parameters onto its CKM. The leftmost two columns on the CKM are reserved (one column for constant operations, one as a "spacer" row with no assignments), leaving six columns of knob cells available for plug-in parameters. S6L uses the 'Av46' 4x6 page table type to map plug-in parameters to the CKM knob cells in this mode.

If a 'Av46' page table is not available from the plug-in, S6L uses the plug-in's C|24 'FrTL' layout in order to map one plug-in parameter to each of the 24 available knob cells. This generally leads to a very sub-optimal layout of parameters on the surface, both because the C|24 page table is designed for a linear layout and because it results in only one parameter assigned per knob cell, leaving two of the available encoders unassigned. Therefore, Avid strongly recommends that all AAX DSP plug-ins which are compatible with S6L are updated to include the new 4x6 page table layout.

S6L also supports dedicated EQ and dynamics plug-ins: a user can select a particular plug-in as his EQ or dynamics processor and use the surface's EQ, Compressor/Limiter, and Expander/Gate selectors to display the plug-in's parameters using a fixed layout for the given plug-in type. This mode uses the plug-in's D-Control Center Section EQ or Dynamics page tables when mapping plug-in controls. In order to be selectable as the system's EQ or dynamics processor a plug-in must meet the criteria for an ICON center-section plug-in.

12.43.3.3.2 VENUE | S3L-X The VENUE | S3L-X System is a modular live sound solution including an HDX-powered processing engine, scalable remote I/O, and an [S3](#) control surface VENUE | S3L-X System

When used as part of an S3L system, the top-right eight encoder/OLED pairs on the S3 control surface (the Global Control encoders) may be placed into Insert mode in order to map to any plug-in instance. When in Insert Mode, the Global Control encoders use the plug-in's 'PcTL' page tables to map parameters onto the surface encoders.

Like S6L, S3L also includes support for dedicated EQ and dynamics plug-ins: a user can select a particular plug-in as his EQ or dynamics processor and use the surface's EQ, Compressor/Limiter, and Expander/Gate selectors to bring up the plug-in on the top-left eight encoder/OLED pairs on the S3 (the Channel Control encoders.) This mode uses the plug-in's D-Control Center Section EQ or Dynamics page tables when mapping plug-in controls. In order to be selectable as the system's EQ or dynamics processor a plug-in must meet the criteria for an ICON center-section plug-in. See [Using Channel Control](#) in the [VENUE Guide](#) and also [Center Section Parameter Mapping on VENUE | S3L-X](#) below for more information about encoder mapping in Channel Control mode.

12.43.4 Plug-In Page Table Guidelines

This section is intended as a guide in setting up defined 'pages' using some general rules. However, due to the sheer number of variables, it is simply not possible to account for all scenarios. But by following these suggestions, an AAX plug-in developer should find these guidelines useful in setting up their own plug-in pages. Moreover, it is hoped that a consistent and somewhat standard mapping topology will be realized across the broad range of plug-ins and control surfaces.

Here, we are primarily concerned with the number of controls on a control surface (CS) available for plug-in editing; and secondarily with the layout of controls provided for plug-ins. Accordingly, we need a method of mapping a plug-in's control parameters to a CS, and we need to take into account the varying numbers of controls available on a CS. Using 'page tables', from a software point of view, a plugin's control parameters can be mapped to a CS. Each page of the page table describes which of the plugin's parameters will be accessible from the CS's controls that are used for plug-in editing. Multiple pages are needed in the case where a CS has fewer controls available than the actual number of controls on a plug-in. We begin by stating guidelines that should be followed when mapping a plug-in's control parameters to a CS. At the end of this chapter, you will find the technical details of creating page tables for your plug-in.

The following guidelines are for simple, generic control surfaces. More advanced CSs, such as the MackieHUI and ProControl, have some guidelines of their own which are listed after these.

12.43.4.1 General Guidelines

Map a plug-in's controls from left-top to right-bottom sequentially onto each page. Follow the layout of the plug-in GUI as closely as possible, allowing the controls to sequentially map to the Control Surface in the order specified above. In so doing, the CS controls will match the plug-in GUI; in the sense that by counting the location of a given control on the PI GUI, one should be able to grasp the corresponding slider or pot on the CS.

Note that Master Bypass, located in the plug-in's floating inserts window, should nearly always be placed as the first control on the first page. The only time this guideline might not be followed is if the plug-in has a particularly favorable layout for the control surface, and where this placement of Master Bypass would disrupt it. Also, on some control surfaces a dedicated bypass is already provided, in which case the Master Bypass should not be mapped into the page table.

Related control parameters should be grouped together on the same page. Controls that are often 'adjusted' with other similar or related controls should be mapped to the same page. This enhances the users ability to tweak related parameters and alleviates unnecessary paging.

Related control parameters should not be split across pages. This follows directly from the bullet above. If some closely related controls cannot all fit on the same page, it is better to leave some blanks (i.e., unused pots, sliders, or switches) and move onto the next page where they can be adjusted together.

As a hypothetical example, let's say a control surface has 5 sliders, and we are mapping an EQ PI with 6 parameters - a low, mid, and high frequency band which has gain for each band. It would be best to map them to the CS as follows on page 1, from left to right on the CS: low freq, low gain, mid freq, mid gain, blank. Then map the remaining two parameters onto page 2: high freq, high gain, blank, blank, blank.

Equivalent left and right stereo parameters should remain on the same page. Since adjusting the left or right parameter of a stereo PI has considerable impact on the sound field, it is important that equivalent left and right stereo controls remain on the same page. Contrast this to placing the left parameters on one page, and the right parameters on another which is not desirable. This rule also changes according to the controller's layout. As an example, the CS-10 has 6 pots for PI editing arranged in a matrix of 3 rows x 2 columns. From left to right, the pots in row 1 are numbered 1 and 4. In row 2 the pots are numbered 2 and 5. Finally, the pots in row 3 are numbered 3 and 6. A layout for L/R controls should be mapped param1L = control 1, param1R = control 4, and so on.

Repeat control parameters on pages where it makes sense to do so. In some situations, it is desirable to have access to the same control on many pages. For example, this might mean having an output and/or input gain control available on each page of an EQ PI - since EQs change the overall gain. This is especially desirable if there would otherwise be blanks (i.e., unused pots, sliders, or switches).

12.43.5 Avid Center Section Page Tables

"Center Section" page tables provide a mapping of plug-in parameters to dedicated functions. These page tables are used by D-Control/D-Command (ICON), D-Show (VENUE), and EUCON-enabled consoles to provide a consistent user experience when interacting with EQ and Dynamics plug-ins.

There are three Center Section page table types:

Table type	Plug-in category
'DgEQ '	EQ
'DgCP '	Compressor/Limiter (Dynamics)
'DgGT '	Expander/Gate (Dynamics)

Dynamics plug-ins that include both Compressor/Limiter and Expander/Gate processing should support both DgCP and DgGT page tables.

The control surfaces which use these page tables each use a different physical layout of parameter functions onto the surface. These layouts have been designed to provide an intuitive and consistent way to control EQ and Dynamics plug-ins in a way that is appropriate for the encoder layout on each surface. By adding these page tables to your EQ and Dynamics plug-ins, your plug-ins will map correctly to all products which use these tables.

12.43.5.1 Center Section Page Table Guidelines

It is important to note that Center Section page table types are different from all other page table types in a fundamental way:

Each slot in the page table is *pre-defined* for a specific type of control. Therefore, your plug-in must conform to this pre-defined layout.

The purpose of these tables is to give the user a standard interface for EQ and Dynamics plug-ins - no matter what particular plug-in they are using. It allows the user to quickly access the most common controls in their favorite EQ or Dynamics plug-ins. This is different from all other page table types because the only restriction on other page table types is that - for types that have dedicated discrete controls - you cannot place continuous controls in a dedicated switch. However, the user will also know that if there are controls they would like to access that are missing from the Center Section, they can access them through one of the other layouts available on the control surface.

You'll notice looking at the pictures of these sections in D-Control (below) that the only scribble strips are in the center of the section. Unlike the Channel Strip section, there is not a scribble strip to label each control. The control's purpose is physically printed on the control surface. This model of hard-coding "center section" plug-in functions to specific encoders is followed on VENUE systems and on EUCON-enabled control surfaces which use these table types as well. That is why it is imperative that your plug-in conform to the pre-defined layout.

Because of the strict definitions of the layouts, it may mean that 1) not all of the controls for your plug-in can fit in these sections, and that 2) there may be controls your plug-in does not have and therefore are blank in this view. For example, let's say your plug-in is a 10-band EQ that does not have individual Q controls on any of the bands. Such a plug-in will be forced to leave off some of its bands, even though all Q controls specific to the bands on the page table are empty. That is fine as there will be another way for the user to display the plug-in on the control surface that will include all of the controls. For example, on D-Control, a user can also view an EQ or Dynamics plug-in in both the Channel Strip and Custom Fader modes which will display all controls. The important point is this:

Do not assign a parameter to a Center Section page table slot that does not match the parameter's function.

If you do, the parameter's function will be mislabeled and will cause confusion for the user.

You should only implement one page for these Center Section layouts. This is different from the other page table types, where it is expected to implement as many pages as necessary to give access to all controls in the plug-in. In the case of the Center Section layouts, you should only define one page, except in the case when the plug-in has separate controls for each channel (Left, Right, Center, etc.).

More than one page in Center Section layouts is allowed *only* if the plug-in has separate controls for each channel.

For example, if your EQ plug-in allows the user to change the EQ differently for the left and right channels, then you would implement two pages for the DgEQ page table. You'll notice in the pictures below for the EQ and Dynamics sections of D-Control, there are buttons labeled for channel selections (L, LC, C, RC, R, etc.). The control surface will automatically map the pages to the buttons, according to the standard order for surround channels. For example, in a stereo EQ, Left controls should be in the first page, and Right controls in the second. D-Control will automatically map page 1 to the L button and page 2 to the R button.

Note

We cannot emphasize strongly enough that trying to fill in all of your controls into these layouts, whether it is by creating extra pages, or by filling in empty slots, will only serve to confuse the user. You must adhere strictly to the guidelines given for these sections.

12.43.5.1.1 EQ Center Section Page Table Guidelines To demonstrate the guidelines for EQ Center Section tables, we will use the D-Control Center Section encoders. Since all control surfaces which use Center Section page tables use a similar approach with hard-coded layouts for these types, the guidelines in this section are equally applicable to all control surfaces.

Take a look at D-Control's EQ section more closely in the image below.

D-Control EQ Center Section.

It supports a maximum of seven bands of EQ, each a vertical column of controls and labeled from left to right as follows:

- HPF (High-Pass Filter, nominally a high-pass filter or low frequency notch filter)
- LF (Low Frequency, nominally a parametric EQ or low frequency shelf filter)
- LMF (Low-Mid Frequency, nominally a parametric EQ)
- MF (Mid Frequency, nominally a parametric EQ)
- HMF (High-Mid Frequency, nominally a parametric EQ)
- HF (High Frequency, nominally a parametric EQ or high frequency shelf filter)
- LPF (Low-Pass Filter, nominally a low-pass filter or high frequency notch filter)

Each of these bands has a Q/Slope control, Frequency control, and an In Circuit/ Out of Circuit button. Five of the bands have an additional Gain control. Four of the bands have an additional EQ type selector switch, each surrounded by a pair of EQ type LED's. Input and Output Level controls are also available, as is a multi-channel Link button in the middle section.

Note

If an EQ plug-in implements a band that does not have an In/Out Circuit control or a Type control, but wants the related LED's to light properly, please see the discussion of the `GetParameterValueInfo()` API below.

The topmost rotary encoders in the HPF, LF, HF, and LPF bands are not labeled but they are indeed Q / Slope controls. The EQ type selector switches located between the Q/Slope and Frequency knobs control the type of filter on that band. Thus they define the behavior of all controls in that band (and not just the unlabeled Q/Slope control).

The EQ page table indices map to the dedicated EQ Center Section hardware encoders as follows:

- Equalization 'DgEQ'
 1. High Pass - In Circuit switch
 2. High Pass - Type switch
 3. High Pass - Frequency
 4. High Pass - Q/Slope
 5. Low Filter - In Circuit switch
 6. Low Filter - Type switch
 7. Low Filter - Gain
 8. Low Filter - Frequency
 9. Low Filter - Q/Slope
 10. Low-Mid Filter - In Circuit switch
 11. Low-Mid Filter - Type switch
 12. Low-Mid Filter - Gain
 13. Low-Mid Filter - Frequency
 14. Low-Mid Filter - Q/Slope
 15. Mid Filter - In Circuit switch
 16. Mid Filter - Type switch
 17. Mid Filter - Gain
 18. Mid Filter - Frequency
 19. Mid Filter - Q/Slope
 20. High-Mid Filter - In Circuit switch

21. High-Mid Filter - Type switch
22. High-Mid Filter - Gain
23. High-Mid Filter - Frequency
24. High-Mid Filter - Q/Slope
25. High Filter - In Circuit switch
26. High Filter - Type switch
27. High Filter - Gain
28. High Filter - Frequency
29. High Filter - Q/Slope
30. Low Pass - In Circuit switch
31. Low Pass - Type switch
32. Low Pass - Frequency
33. Low Pass - Q/Slope
34. Input Gain
35. Output Gain
36. Multi-channel Link switch
37. High Pass - Q/Slope alternate parameter
38. Low Filter - Q/Slope alternate parameter
39. Low-Mid Filter - Q/Slope alternate parameter
40. Mid Filter - Q/Slope alternate parameter
41. High-Mid Filter - Q/Slope alternate parameter
42. High Filter - Q/Slope alternate parameter
43. Low Pass - Q/Slope alternate parameter

Note

In order to use the "Q/Slope alternate parameter" functions in the EQ page table, a plug-in must respond to the [AAX_ePageTable_UseAlternateControl](#) selector in the [GetParameterValueInfo\(\)](#) method. When the band type is changed, the control surface will call into the plug-in with this selector to determine if the control in the "Alt" position should be used. Please see the discussion of the [GetParameterValueInfo\(\)](#) below.

If your plug-in supports fewer than seven simultaneous bands of EQ, you have some options for where to place them. We recommend the following placement guidelines so users have consistency with various EQ plug-ins.

- If you only have one band of EQ, use the LF band.
- If you have one to four fully parametric bands, use LF, LMF, HMF, and HF (starting from left to right). (Skip the MF band.)
- If you have up to two shelving filters and two parametric bands, use LF (LF shelf), LMF (para), HMF (para), and HF (HF shelf). (Skip the MF band.)

Note that this layout includes a few additional functions not in D-Control's EQ section. These extra functions are the "Low-Mid Filter - Type switch", "Mid Filter - Type switch", and "High-Mid Filter - Type switch" slots.

12.43.5.1.2 Dynamics Center Section Page Table Guidelines To demonstrate the guidelines for Dynamics Center Section tables, we will use the D-Control Center Section encoders. As with the EQ Center Section tables above, all control surfaces which use Center Section page tables use a similar approach with hard-coded layouts for these types, the guidelines in this section are equally applicable to all control surfaces.

The D-Control Dynamics section is shown below. Keep in mind that the D-Control's Dynamics section displays page tables for both the Compressor/Limiter page table type and the Expander/Gate type. Therefore, the function of certain rotaries and switches differs depending on which page table type has been loaded. In these cases, there is a LED to indicate the current function of a rotary or switch.

D-Control Dynamics Center Section.

Several rotary encoders have alternate uses while others are always dedicated to one function. The two page tables' indices map to the dedicated Dynamics Center Section hardware encoders as follows:

- Compressor/Limiter 'DgCP'
 1. Threshold
 2. Ratio
 3. Attack Time
 4. Release Time
 5. Knee
 6. Make-up Gain
 7. High Pass - In Circuit / Out of Circuit switch
 8. High Pass - EQ Type switch (with notch and high-pass filter LEDs)
 9. High Pass - Frequency
 10. High Pass - Q/Slope
 11. Low Pass - In Circuit / Out of Circuit switch
 12. Low Pass - EQ Type switch (with notch and low-pass filter LEDs)
 13. Low Pass - Frequency
 14. Low Pass - Q/Slope
 15. External Key switch (middle section)
 16. Key Listen switch (middle section)
 17. Input Gain
 18. Output Gain
 19. Multi-channel Link switch (middle section)
 20. Depth (not included on ICON)
- Expander/Gate 'DgGT'
 1. Threshold
 2. Ratio
 3. Range
 4. Attack Time
 5. Release Time
 6. Hysteresis
 7. Hold
 8. High Pass - In Circuit / Out of Circuit switch
 9. High Pass - EQ Type switch (with notch and high-pass filter LEDs)
 10. High Pass - Frequency

11. High Pass - Q/Slope
12. Low Pass - In Circuit / Out of Circuit switch
13. Low Pass - EQ Type switch (with notch and low-pass filter LEDs)
14. Low Pass - Frequency
15. Low Pass - Q/Slope
16. External Key switch (middle section)
17. Key Listen switch (middle section)
18. Input Gain
19. Output Gain
20. Multi-channel Link switch (middle section)
21. Knee (not included on ICON)

The compressor/limiter page table, DgCP, supports all the controls above except Range, Hysteresis, and Hold. (As you create the page table in the [Page Table Editor](#), this is clear.)

The expander/gate page table, DgGT, supports all the controls above except Knee and Makeup Gain. It does support both Ratio and Range. The user presses the Page button to select between them.

A plug-in can support both DgCP and DgGT page tables. Again, the user presses the Page button to select between them. If a plug-in supports both of these page tables and the DgGT page table includes support for both Ratio and Range controls, pressing the Page button will switch between all available controls as follows:

DgCP -> DgGT with Ratio -> DgGT with Range (and all other controls unchanged)-> DgCP -> ...

12.43.5.2 Center Section Parameter Mapping to Single-Column/Row Layouts

The following tables show the layout mapping and hard-coded names for center section page tables on EUCON surfaces which use single-column or single-row layouts for Eq and Dyn plug-in modes. The tables show the assignment of specific center section table indices to cells on the EUCON control surface. Each EUCON control surface cell includes three encoders: a rotary knob encoder and two push-button encoders. These tables use the following codes to indicate encoders in each control surface cell:

- Knob = Knob encoder
- Knob(Sel I) = Knob encoder with Sel active
- Knob(Sel O) = Knob encoder with Sel inactive
- In = In switch

Note

For center section page table layouts the "Sel" push-button encoder is only used to toggle the rotary encoder between two possible parameters. It is never mapped to a single discrete parameter in these layouts.

With some versions of EUCON, the cell mapping on the first page is "reversed" relative to the second page when the surface is laid out horizontally; the first page moves left to right through increasing cell indices, while later pages move right to left.

		Page 1	Page 2
--	--	--------	--------

	Function	Page 1								Page 2							
		Far from user / Left				Close to user / Right				Far from user / Left				Close to user / Right			
		8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9
1	HPF In/↔ Out														Knob/In		
2	HPF Type													In			
3	HPF Freq													Knob (Sel O)			
4	HPF Q													Knob (Sel I)			
5	Lo In/↔ Out							In									
6	Lo Type								In								
7	Lo Gain							Knob									
8	Lo Freq								Knob (Sel O)								
9	Lo Q								Knob (Sel I)								
10	Lo Mid In/↔ Out					In											
11	Lo Mid Type						In										
12	Lo Mid Gain					Knob											
13	Lo Mid Freq						Knob (Sel O)										

		Page 1						Page 2					
14	Lo Mid Q					Knob (Sel I)							
15	Mid In/↔ Out										In		
16	Mid Type									In			
17	Mid Gain										Knob		
18	Mid Freq									Knob (Sel O)			
19	Mid Q									Knob (Sel I)			
20	Hi Mid In/↔ Out			In									
21	Hi Mid Type				In								
22	Hi Mid Gain			Knob									
23	Hi Mid Freq				Knob (Sel O)								
24	Hi Mid Q				Knob (Sel I)								
25	Hi In/↔ Out	In											
26	Hi Type		In										
27	Hi Gain	Knob											
28	Hi Freq		Knob (Sel O)										
29	Hi Q		Knob (Sel I)										
30	LPF In/↔ Out								Knob/In				

		Page 1							Page 2						
31	LPF Type LPF Freq LPF Q In- put Level Out- put Level								In						
32									Knob (Sel O)						
33									Knob (Sel I)						
34														Knob	
35															Knob
36															
37	Link														
38	HPF Q Alt												Knob (Sel I)*		
39	Lo Q Alt							Knob (Sel I)*							
40	Lo Mid Q Alt							Knob (Sel I)*							
41	Mid Q Alt								Knob (Sel I)*						
42	Hi Mid Q Alt														
43	Hi Q Alt														
44	LPF Q Alt								Knob (Sel I)*						

12.43.5.2.1 'DgEQ' PageTable - Equalizer Notes

- The multi-channel link switch (index 36) is not mapped to any encoder in this layout
- The knob assignments marked with an asterisk will be assigned depending on the plug-in's response to [AAX_IEffectParameters::GetParameterValueInfo\(\)](#) with the [AAX_ePageTable_UseAlternateControl](#) selector. If the plug-in provides [AAX_eUseAlternateControl_Yes](#) then the assignment marked with an asterisk will be used.

12.43.5.2.2 Both 'DgCP' and 'DgGT' PageTables - Multi-dynamics If both Dynamics center section page table types are defined for the plug-in then EUCON control surfaces will use the following mapping.

Note

Some control surfaces may only partially follow this mapping. For example, the mapping of the Artist Mix control surface in Dyn mode uses two pages: it follows this mapping for encoder cells 1-8 on the first page and follows the ['DgCP'-only mapping](#) for encoder cells 9-16 on the second page.

		Page 1								Page 2							
		Far from user / Left				Close to user / Right				Close to user / Left				Far from user / Right			
		8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9
Dg←CP 1	Function C Threshold						Knob				Knob						
Dg←CP 2	Function C Ratio					Knob				Knob							
Dg←CP 3	Function C Attack Time							Knob					Knob				
Dg←CP 4	Function C Release Time								Knob					Knob			
Dg←CP 5	Function C Knee											Knob					
Dg←CP 6	Function C Gain Makeup															Knob	
Dg←CP 7	HPF Enabled																
Dg←CP 8	HPF Type																
Dg←CP 9	HPF Freq																
Dg←CP 10	HPF Q																
Dg←CP 11	LPF Enabled																

		Page 1								Page 2							
Dg← CP 12	LPF Type																
Dg← CP 13	LPF Freq																
Dg← CP 14	LPF Q																
Dg← CP 15	Ext Key																
Dg← CP 16	Key Lis- ten																
Dg← CP 17	In- put Gain																
Dg← CP 18	Out- put Gain																
Dg← CP 19	Link														Knob		
Dg← CP 20	Depth																
Dg← GT 1	X Thresh- old		Knob														
Dg← GT 2	X Ra- tio		Knob														
Dg← GT 3	X Range																
Dg← GT 4	X At- tack Time			Knob													
Dg← GT 5	X Re- lease Time				Knob												

		Page 1								Page 2							
Dg← GT 6	X Hys- tere- sis																
Dg← GT 7	X Hold																

		Page 3								Page 4							
		Close to user / Left				Far from user / Right				Close to user / Left				Far from user / Right			
		24	23	22	21	20	19	18	17	32	31	30	29	28	27	26	25
	Function																
Dg← CP 1	C Thresh- old																
Dg← CP 2	C Ra- tio																
Dg← CP 3	C At- tack Time																
Dg← CP 4	C Re- lease Time																
Dg← CP 5	C Knee																
Dg← CP 6	C Gain Makeup																
Dg← CP 7	HPF En- abled														Knob/In		
Dg← CP 8	HPF Type													In			
Dg← CP 9	HPF Freq													Knob (Sel O)			
Dg← CP 10	HPF Q													Knob (Sel I)			

		Page 3							Page 4						
Dg← CP 11	LPF En- abled											Knob/In			
Dg← CP 12	LPF Type										In				
Dg← CP 13	LPF Freq										Knob (Sel O)				
Dg← CP 14	LPF Q										Knob (Sel I)				
Dg← CP 15	Ext Key									Knob/In					
Dg← CP 16	Key Lis- ten								Knob/In						
Dg← CP 17	In- put Gain												Knob		
Dg← CP 18	Out- put Gain													Knob	
Dg← CP 19	Link														
Dg← CP 20	Depth														
Dg← GT 1	X Thresh- old		Knob												
Dg← GT 2	X Ra- tio		Knob												
Dg← GT 3	X Range					Knob									
Dg← GT 4	X At- tack Time			Knob											

		Page 3								Page 4							
Dg← GT 5	X Re- lease Time					Knob											
Dg← GT 6	X Hys- tere- sis								Knob								
Dg← GT 7	X Hold							Knob									

Notes

- Neither table's multi-channel link switch ('DgCP ' index 19 and 'DgGT ' index 20) is mapped to an encoder in this layout
- The 'DgGT ' filter, external key, and gain parameters (indices 8 through 19) are not mapped to any encoders in this layout

12.43.5.2.3 'DgCP' PageTable - Compressor/Limiter If the plug-in defines a 'DgCP ' page table but does not define a 'DgGT ' page table then EUCON control surfaces will use the following mapping.

		Page 1*								Page 2							
		Far from user / Left				Close to user / Right				Far from user / Left				Close to user / Right			
		8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9
1	C Thresh- old						Knob										
2	C Ra- tio					Knob											
3	C At- tack Time							Knob									
4	C Re- lease Time								Knob								
5	C Knee	Knob															
6	C Gain Makeup		Knob*			Knob*											
7	HPF En- abled											Knob//In					

8	HPF Type	Page 1*								Page 2							
													In				
9	HPF Freq												Knob (Sel O)				
10	HPF Q												Knob (Sel I)				
11	LPF Enabled												Knob/In				
12	LPF Type													In			
13	LPF Freq													Knob (Sel O)			
14	LPF Q													Knob (Sel I)			
15	Ext Key															Knob/In	
16	Key Listen																Knob/In
17	Input Gain			Knob													
18	Output Gain				Knob												
19	Link																
20	Depth									Knob							

Notes

- If no parameter is defined at the Ratio parameter index then the Makeup Gain parameter will be mapped to the Ratio parameter's normal spot in order to increase usefulness of the first-page mapping.
- Pro Tools versions prior to Pro Tools 11.1 use a different layout for the first page of this table type

12.43.5.2.4 'DgGT' PageTable - Expander/Gate If the plug-in defines a 'DgGT' page table but does not define a 'DgCP' page table then EUCON control surfaces will use the following mapping.

		Page 1								Page 2							
		Far from user / Left				Close to user / Right				Far from user / Left				Close to user / Right			
		8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9
1	Function X Threshold							Knob									
2	Function X Ratio								Knob								
3	Function X Range			Knob													
4	Function X Attack Time					Knob											
5	Function X Release Time				Knob												
6	Function X Hysteresis	Knob															
7	Function X Hold		Knob														
8	HPF Enabled											Knob/In					
9	HPF Type												In				
10	HPF Freq												Knob (Sel O)				
11	HPF Q												Knob (Sel I)				
12	LPF Enabled													Knob/In			
13	LPF Type														In		
14	LPF Freq														Knob (Sel O)		
15	LPF Q														Knob (Sel I)		
16	Ext Key															Knob/In	

17	Key Listen	Page 1								Page 2							
																	Knob/In
18	In-put Gain									Knob							
19	Out-put Gain									Knob							
20	Link																

12.43.5.3 Center Section Parameter Mapping in S6 Expand Mode

In addition to supporting single-row and single-column EQ and Dynamics layouts as described above, S6 also includes an Expand Mode for EQ and Dynamics plug-ins which allows the targeted plug-in's EQ or Dynamics center section mapping to be displayed across an entire CKM module.

12.43.5.3.1 'DgEQ' PageTable - Equalizer EQ layout in S6 Expand Mode

12.43.5.3.2 'DgCP' and 'DgGT' PageTables - Compressor/Limiter and Expander/Gate Dynamics layout in S6 Expand Mode

12.43.5.4 Center Section Parameter Mapping on VENUE | S3L-X

Built-in processing parameters are mapped to the Channel Control encoders on VENUE | S3L. For more information about Channel Control mode in S3L-X see [Using Channel Control](#) in the [VENUE Guide](#).

Channel Control Encoder	Knob	Sel Switch	In Switch
1	Low Gain		Low EQ band in/out
2	Low Freq/Q	Toggles Freq/Q	Low EQ band type
3	LoMid Gain		LoMid EQ band in/out
4	LoMid Freq/Q	Toggles Freq/Q	
5	HiMid Gain		HiMid EQ band in/out
6	HiMid Freq/Q	Toggles Freq/Q	
7	High Gain		High EQ band in/out
8	High Freq/Q	Toggles Freq/Q	High EQ band type

12.43.5.4.1 'DgEQ' PageTable - Equalizer

Channel Control Encoder	Knob	Sel Switch	In Switch
1	Threshold level		Comp/Lim or Exp/Gate in/out
2	Ratio		
3	Attack		
4	Knee		
5	Release		
6	Gain level		
7	Key HF	Key Listen	Filter in/out
8	Key LF	Key In	Filter in/out

12.43.5.4.2 'DgCP' and 'DgGT' PageTables - Compressor/Limiter and Expander/Gate

12.43.6 EUCON Page Tables

Plug-ins should implement specific EUCON page tables to take advantage of EUCON-specific features and layouts. By writing EUCON-specific page tables, your plug-in is able to re-define both the in/out button as well as the select button per EUCON control cell on compatible surfaces.

Host Compatibility Notes Pro Tools versions prior to Pro Tools 11.1 use plug-ins' ProControl and ICON page tables (Dynamics, EQ, Channel Strip, Custom Fader, etc.) to map plug-in parameters to EUCON-enabled surfaces, so be sure that your plug-ins also implement these page tables correctly so that users with earlier versions of Pro Tools can have the best possible experience when using your plug-ins.

Note

The legacy PeTE editor will remove all EUCON sections from any page table XML file that it saves. Developers should no longer use PeTE for [AAX](#) page table editing. If you do use Pete, it is important to *always back up your page table file* before editing the file in PeTE.

12.43.6.1 Specification

EUCON page tables use modern formatting, making them more intuitive to implement than non-EUCON tables. Here are some example lines from a EUCON page table:

```
<PageTable type='Av18' pgsz='24' >
  <Page num='1'\>
    <Cell row='1' col='1' knobID="Knob1" inOutButtonID="Button1A" selectButtonID="Button1B" />
    <Cell row='1' col='2' knobID="Knob2" inOutButtonID="Button2A" selectButtonID="Button2B" />
    <Cell row='1' col='3' knobID="Knob3" inOutButtonID="Button3A" selectButtonID="Button3B" />
    ...
  </Page\>
  ...
</PageTable >
```

The EUCON PageTable element includes a series of Cell sub-elements. The attributes of each Cell sub-element are as follows:

- **row** - the cell's row position, ordered furthest to nearest
- **col** - the cell's column position, ordered left to right
- **knobID** - the parameter ID associated with the knob in question
- **inOutButtonID** - the parameter ID associated with the "In" button next to the knob in question. This must be a discrete parameter.
- **selectButtonID** - the parameter ID associated with the "Sel" button next to the knob in question. This can be a discrete or continuous parameter. If it is a continuous parameter then pushing "Sel" will toggle the knob associated with the button between the selectButtonID and knobID parameters.

12.43.6.2 Types

Av81	Av18	Av48	Av46
type='Av81'	type='Av18'	type='Av48'	type='Av46'
8 rows	1 row	4 rows	4 rows
1 column	8 columns	8 columns	6 columns
pgsz='24' (3 elements per cell)	pgsz='24' (3 elements per cell)	pgsz='96' (3 elements per cell)	pgsz='72' (3 elements per cell)
Used for vertical sets of knob cells	Used for horizontal sets of knob cells	Used for 4x8 knob cell arrays	Used for 6x4 knob cell arrays, e.g. plug-in layout
Most important parameters should be placed in this table	Most important parameters should be placed in this table	Most important parameters should be placed in this table	Most important parameters should be placed in this table

12.43.6.3 Conventions

- To map a single discrete parameter to an encoder cell, assign the parameter to both the knob and the In switch. Assigning the parameter to the cell's knob will ensure that the parameter name and value is always displayed on the surface. The user will be able to edit the parameter using either the rotary encoder or the In switch.
- When assigning discrete parameters, always prefer to use the In switch over the Sel switch. For cells with one continuous and one discrete parameter, users will always expect the discrete parameter to be assigned to the In switch rather than the Sel switch. In other EUCON modes (besides plug-in editing) the In switch is always used to enable/disable parameters, while the Sel switch is usually used to toggle between functions for the cell's rotary encoder.

12.43.6.4 Requirements

- All parameter IDs used in the EUCON page tables must be defined with a `Ctrl ID` element within the `ControlNameVariations` element
- Every `Cell` with at least one parameter assignment must include a `knobID` assignment. It is not valid to assign either `inOutButtonID` or `selectButtonID` without also assigning the cell's `knobID`.
- For a given `knobID`, the same parameters must be assigned to the `selectButtonID` and `inOutButtonID` switches across all EUCON page tables
- Every knob cell assignment set (Rotary+Sel+In assignment) used in the 'Av48' table must be exactly replicated somewhere in the 'Av81' table
- 'Av48' tables may contain no more than two pages

12.43.7 Implementing Page Tables

12.43.7.1 Page table XML specification

This section includes a rough specification for plug-in page table XML. Whenever possible, we encourage developers to use the [Page Table Editor](#) tool to generate plug-in page tables rather than writing or editing the page table XML by hand.

The page table XML format contains three main tags:

- [PageTableLayouts](#) tag
- [ControlNameVariations](#) tag
- [Editor](#) tag

12.43.7.1.1 PageTableLayouts tag This section provides a static mapping of control elements to page table layouts. Multiple layouts may be provided in this section, e.g. in cases where different control sets are used by different plug-in Types.

Each layout includes a complete set of page table descriptions. There are multiple kinds of page tables, each of which may have multiple pages. At minimum, each layout must include a PageTable with type='PgTL' and pgsz='1'. This is the default page table, and the order of the control elements that it describes must match the order in which the corresponding parameters are added to the plug-in itself.

Here is an excerpt from the PageTableLayouts section in Avid's Eleven plug-in page tables demonstrating non-EUCON PageTable elements. For the EUCON PageTable element specification, see [Eucon page table specification](#).

```
<PageTableLayouts>
  <Plugin manID='Digi' prodID='ElvF' plugID='ELFr'>
    <Desc>Eleven Free 1 -&gt; 1 Avid Technology, Inc.</Desc>
    <Layout>PageTable Free</Layout>
  </Plugin><!--manID='Digi' prodID='ElvF' plugID='ELFr'-->
  <Plugin manID='Digi' prodID='Elvn' plugID='ELVr'>
    <Desc>Eleven 1 -&gt; 1 Avid Technology, Inc.</Desc>
    <Layout>PageTable 1</Layout>
  </Plugin><!--manID='Digi' prodID='Elvn' plugID='ELVr'-->

  <!-- ... -->

  <PTLayout name='PageTable 1'>
    <PageTable type='BkCS' pgsz='12'>
      <Page num='1'>
        <ID>Mic Type</ID>
        <ID>Cab Type</ID>
        <ID>Amp Type</ID>
        <ID>Master</ID>
        <ID>Gain 2</ID>
        <ID>Gain 1</ID>
        <ID>Mic Axis</ID>
        <ID>Cab Type</ID>
        <ID>Amp Type</ID>
        <ID> </ID>
        <ID> </ID>
        <ID>Bright Switch</ID>
      </Page><!--num='1'-->
      <Page num='2'>
        <!-- ... -->
      </Page><!--num='2'-->
      <Page num='3'>
        <!-- ... -->
      </Page><!--num='3'-->
    </PageTable><!--type='BkCS' pgsz='12'-->

    <!-- ... -->

    <PageTable type='PgTL' pgsz='1'>
      <Page num='1'>
        <ID>Master Bypass</ID>
      </Page><!--num='1'-->
      <Page num='2'>
        <ID>Input Level</ID>
      </Page><!--num='2'-->
      <Page num='3'>
        <ID>Output Level</ID>
      </Page><!--num='3'-->
      <Page num='4'>
        <ID>Gate Threshold</ID>
      </Page><!--num='4'-->

      <!-- ... -->

    </PageTable><!--type='PgTL' pgsz='1'-->
  </PTLayout><!--name='PageTable 1'-->
  <PTLayout name='PageTable Free'>
    <!-- ... -->
  </PTLayout><!--name='PageTable Free'-->
</PageTableLayouts>
```

The sub-tags for this section are as follows:

- Plugin element

A high-level description of a single plug-in Type.

The `manID`, `prodID`, and `plugID` attributes must match the corresponding Manufacturer, Product, and Type IDs for each plug-in Type that is described in the XML file. Multiple `Plugin` elements may be included in a single XML file.

The `Plugin` element has two sub-elements:

1. The `Desc` sub-element provides a brief description of the plug-in Type. This information is used only by the [Page Table Editor](#) application and does not affect operation of the plug-in in Pro Tools.
2. One `Layout` sub-element is used to bind the plug-in Type to a particular `PTLayout` (see below.)

- `PTLayout` element

A complete control mapping, including a full set of `PageTable` descriptions.

- `PageTable` sub-element - A single page table mapping, with controls specified across multiple `Page` elements as in the example above.

`PageTable` elements have the following attributes:

1. `type` defines the particular device that will use the `PageTable`. `type` may be one of:
 - * `PgTL` - Generic page tables (any size)
 - * `PcTL` - ProControl, VENUE, and fall-back EUCON page table (size 16)
 - * `MkTL` - Makie HUI page table (size 8)
 - * `HgTL` - 002/003 and Command|8 page table (size 8)
 - * `FrTL` - Control 24, C|24, and fall-back S6L page table (size 24)
 - * `BkCS` - ICON Channel Strip (size 12)
 - * `BkSF` - ICON Custom Fader (size 16)
 - * `DgGT` - ICON dynamics section (Gate/Expander) (size 20)
 - * `DgCP` - ICON dynamics section (Compressor/Limiter) (size 19)
 - * `DgEQ` - ICON EQ section (size 43)
 - * `Av81` - EUCON 8x1 section (size 24)
 - * `Av18` - EUCON 1x8 section (size 24)
 - * `Av48` - EUCON 4x8 section (size 96)
 - * `Av46` - EUCON 4x6 section (size 72)
2. `pgsz` defines the number of controls per page in the page table. Most page table types require a specific size, as noted above. The generic `PgTL` page tables may be of any size, and multiple `PgTL` page tables may be provided. However, each plug-in must provide a `PgTL` page table of size 1 that includes all of the plug-in's automatable parameters, in the order in which they are added to the plug-in.

12.43.7.1.2 ControlNameVariations tag This section includes information about the names of each control in the plug-in. Here is an excerpt from our Eleven plug-in page tables which demonstrates the basic format of this section:

```
<Ctrl ID='Amp Bypass'>
  <name typ='PgTL' sz='1'>AB</name>
  <name typ='PgTL' sz='4'>AByp</name>
  <name typ='PgTL' sz='5'>A Byp</name>
  <name typ='PgTL' sz='6'>AmpByp</name>
  <name typ='PgTL' sz='7'>Amp Byp</name>
</Ctrl><!--ID='Amp Bypass'-->
<Ctrl ID='Amp Type'>
  <name typ='PgTL' sz='1'>AT</name>
  <name typ='PgTL' sz='3'>Amp </name>
  <name typ='PgTL' sz='5'>AmpTp</name>
  <name typ='PgTL' sz='6'>AmpTyp</name>
  <name typ='PgTL' sz='7'>Amp Typ</name>
</Ctrl><!--ID='Amp Type'-->
<Ctrl ID='Bright Switch'>
  <name typ='PgTL' sz='1'>Brt </name>
  <name typ='PgTL' sz='6'>Bright</name>
</Ctrl><!--ID='Bright Switch'-->
```

The sub-tags used are as follows:

`Ctrl` element with `ID` attribute - The identifier of the control that is being identified.

The identifiers for all parameters in the page table must and should match the IDs used for the control both in the `<PageTableLayouts>` layouts and in the `Editor` tag's `DiscCtrls` sub-tag (see below). See [Parameter identifiers](#) for more information about parameter identifiers.

`name` element - The desired abbreviated name of the control for display on control surface UIs, which may have limited display space available.

The `typ` parameter allows you to choose which page table type the given abbreviation will be specific to. For example, a name provided with `typ='HgTL'` would only appear on Command|8, 002, and 003 hardware. As with all other tags, the name associated with `typ=PgTL` (the generic page table identifier) will be used if no other name is given for the specific page table that is being loaded.

The `sz` parameter defines the size of the control surface display for which the given name is appropriate. The control surface will use the name associated with the largest size that will fit on its display.

To reduce the number of names that must be specified, abbreviated names can be given that are longer than their specified size. These names will be truncated when necessary. For example, a control surface that could accommodate two characters on its display would use the size-1 name for the "Bright Switch" control above, since 1 is the largest number less than or equal to 2 from the provided set of name sizes (in this case, 1 and 6.) The size-1 name, 'Brt ', has four characters in it. Therefore, it would be truncated down to 'Br' to fit onto the control surface's display.

12.43.7.1.3 Editor tag The last of the three main sections in an XML plug-in page table is the `Editor` section. This section is used by the [Page Table Editor](#) application and you should not need to modify its contents. However, if you are encountering problems modifying your plug-in in the Page Table Editor then you may wish to verify that all plug-ins and controls are properly identified within the `PluginList` and `DiscCtrls` sub-tabs, respectively. Here is the `Editor` section from the Eleven page table XML:

```
<Editor vers='1.3.7.1'>
  <PluginList>
    <TDM>
      <PluginID manID='Digi' prodID='Elvn' plugID='ELVt'>
        <MenuStr>TDM: Eleven, 1 in X 1 out</MenuStr>
      </PluginID><!--manID='Digi' prodID='Elvn' plugID='ELVt'-->
    </TDM>
    <RTAS>
      <PluginID manID='Digi' prodID='Elvn' plugID='ELVr'>
        <MenuStr>RTAS: Eleven, 1 in X 1 out</MenuStr>
      </PluginID><!--manID='Digi' prodID='Elvn' plugID='ELVr'-->
      <PluginID manID='Digi' prodID='ElvF' plugID='ELFr'>
        <MenuStr>RTAS: Eleven Free, 1 in X 1 out</MenuStr>
      </PluginID><!--manID='Digi' prodID='ElvF' plugID='ELFr'-->
    </RTAS>
  </PluginList>
  <DiscCtrls>
    <CtrlID>Amp Bypass</CtrlID>
    <CtrlID>Amp Type</CtrlID>
    <CtrlID>Bright Switch</CtrlID>
    <CtrlID>Cab Bypass</CtrlID>
    <CtrlID>Cab Type</CtrlID>
    <CtrlID>Master Bypass</CtrlID>
    <CtrlID>Mic Axis</CtrlID>
    <CtrlID>Mic Type</CtrlID>
  </DiscCtrls>
</Editor><!--vers='1.3.7.1'-->
```

12.43.7.2 Parameter identifiers

The `ID` tags/arguments in a page table must reference parameters which are exposed by the plug-in's [AAX_IEffectParameters](#) implementation via methods such as [GetParameterIndex\(\)](#).

There are two supported ways to identify parameters in a page table:

- By the parameter's ID (preferred)
- By the parameter's 31-character name (legacy)

A single page table may only reference the plug-in's parameters using one of these two approaches; a page table file may not reference one parameter by its name and another by its ID, and it may not reference one parameter by its name in one location but by its ID in another location.

If a plug-in will change its parameters' names at run time then the parameter identifiers used in the page tables must reference parameters by ID.

12.43.7.3 Creating page tables using the AAX Plug-In Page Table Editor

Page tables can be created and edited using the AAX Plug-In Page Table Editor application available on the Developer website. The Page Table Editor generates an XML file that can be used in both Windows and Macintosh plug-in projects.

The Page Table Editor can also open page table files in existing .aaxplugin bundles on your system. If you're looking for examples of how to lay out the common parameters in your plug-ins try opening up the page tables for one of the standard Avid plug-ins like Avid Channel Strip or D-Verb.

Note

The Page Table Editor only supports opening a single file at a time. It can be useful to open multiple files in order to compare their layouts. To open multiple page tables at the same time you can launch multiple instances of the Page Table Editor application by running the application from a terminal shell.

The Page Table Editor will make use of the Parameter IDs that are defined in a plug-in, and will associate them with the 'Plugin' tags in the XML file. Using the DemoDist sample plug-in included in the AAX SDK, you'll see that there is one Plugin entry for each plug-in type in the binary.

```
// Type, product, and relation IDs
const AAX_CTypeID cDemoDist_ManufactureID = 'AVID ' ;
const AAX_CTypeID cDemoDist_ProductID = 'DmDE ' ;
const AAX_CTypeID cDemoDist_TypeID_AS = 'DmAS ' ;
const AAX_CTypeID cDemoDist_TypeID_MonoNative = 'DmRT ' ;
const AAX_CTypeID cDemoDist_TypeID_StereoNative = 'DsRT ' ;
const AAX_CTypeID cDemoDist_TypeID_MonoTI = 'DDT1 ' ;
const AAX_CTypeID cDemoDist_TypeID_StereoTI = 'DDT2 ' ;
```

Listing 1: DemoDist Plug-In IDs

```
.
.
.
<Plugin manID='AVID ' prodID='DmDE ' plugID='DmRT '>
<Desc > DemoDist 1 -&gt; 1 by Avid Technology , Inc .</ Desc >
<Layout > PageTable 1</ Layout >
</ Plugin ><!-- manID='AVID ' prodID='DmDE ' plugID='DmRT '-->
<Plugin manID='AVID ' prodID='DmDE ' plugID='DsRT '>
<Desc > DemoDist 2 -&gt; 2 by Avid Technology , Inc .</ Desc >
<Layout > PageTable 1</ Layout >
</ Plugin ><!-- manID='AVID ' prodID='DmDE ' plugID='DsRT '-->
<Plugin manID='AVID ' prodID='DmDE ' plugID='DDT1 '>
<Desc > DemoDist 1 -&gt; 1 by Avid Technology , Inc .</ Desc >
<Layout > PageTable 1</ Layout >
</ Plugin ><!-- manID='AVID ' prodID='DmDE ' plugID='DDT1 '-->
<Plugin manID='AVID ' prodID='DmDE ' plugID='DDT2 '>
<Desc > DemoDist 2 -&gt; 2 by Avid Technology , Inc .</ Desc >
<Layout > PageTable 1</ Layout >
</ Plugin ><!-- manID='AVID ' prodID='DmDE ' plugID='DDT2 '-->
.
.
.
```

Listing 2: DemoDist XML

Note

For compatibility between your AAX and corresponding RTAS or TDM plug-ins, make sure the 4 character IDs for [AAX_eProperty_ManufacturerID](#), [AAX_eProperty_ProductID](#), [AAX_eProperty_PluginID_Native](#), and [AAX_eProperty_PluginID_AudioSuite](#) are identical to the legacy SDK's counterpart.

For more information about the AAX Plug-In Page Table Editor tool, see the ReadMe file which accompanies the application.

12.43.7.4 Verifying Page Table Layouts: The Hidden Pop-Up Menu

You can verify the page tables created in your plug-in in Pro Tools with a "hidden" developer debug Page Tables popup menu. To verify the page tables, first include the `YourPageTables.r` file in your project and compile the plug-in. Then, after launching Pro Tools, instantiate the plug-in, hold down the Commandkey (Ctrl-key in Windows) and mouse-click the Automation button in the plug-in window to display the menu.

- Category

The Category menu item has a submenu listing the names of possible categories. Any category that the plug-in belongs to will have a check mark next to it. In addition, appended to the name of the category is an indication of whether that category can be bypassed, and if so, the control number (#) and control name of the associated bypass control. Also appended is the number of the first page on which a control associated with the category can be found. This page number is based on whatever the current page table type is selected in the "TableType" menu item. For example: Delay (can bypass, control #9, Master Bypass) (first page #1)

- Table Type

Sets the type of control surface page table.

- Page Size

Sets the number of controls per page. The identifier "custom" is shown next to page sizes that have been specifically implemented (which at minimum, should appear next to page sizes of 5, 6, 8, and 16!). The identifier "default" is shown next to page sizes that do not have specific support in your page table file. Note that the Mackie and ProControl table type will automatically set this to 8 and 16, respectively.

- Control Name Length

Sets the number of characters to be displayed in the control's name (shown in the Page menu below). The identifier "expected length" appears next to lengths that should be specifically addressed (3, 4, 5, 6, 7, 8, and 31). If you use the XML page table system, the names can be specified in the [Page Table Editor](#) application. Otherwise, the function `GetControlNameOfLength()` is responsible for providing names with these lengths.

- Control Value Length

Sets the number of characters to be displayed in the control's value (shown in the Page menu below). The identifier "expected length" appears next to lengths that should be specifically addressed (4, 5, 6, 7, 8, and 31; also, 3 is used for ProControl switch states). The plug-in Library call `GetValueString()` is responsible for providing values with these lengths.

- Highlighted Page

Highlights the selected page in the plug-in window in Pro Tools.

- Highlight Color

Sets the highlight color. The highlight color can be: red, green, blue or yellow. Note that at minimum, plug-in's should support these four colors of highlighting!

- Page X

The actual page table layouts are shown here. The following information can be seen in this menu item.

- The control's name, as returned from the XML page tables. If the table type is Mackie, then the length will be 4 (unless overridden by changing the "Control Name Length" menu item). If the table type is ProControl, the length will be 3 (again, unless overridden, but usually there is no point in doing so). Also, with ProControl set as the table type, the special ProControl symbols will appear here if they are part of the name (however, note that they are small and can be difficult to see). Finally, if the table type is default, the name will be shown with the number of characters as specified in the "Control Name Length" menu item.
- The control's value is shown next. Both the Mackie and ProControl table types will automatically set the control's value length to 4 and 3, respectively. Otherwise, this can be set with the "Control Value Length" menu item. `GetValueString()` is responsible for providing this 'value' information.

- The number of control steps is shown next for continuous and discrete controls. For example, a discrete control will appear as: "(discrete: N steps)", where N is the # of steps of the control.
- If "(NoL)" is displayed, this simply means that it is a new plug-in, and supports either the XML page tables or the `GetParameterNameOfLength()` function call. If "(NoL)" does not appear, then the plug-in is older and you should not expect the control names or values to be optimized - since they are created by just truncating the longer values that the older plug-ins return.
- If "(highlight)" is displayed, this means that the string will be reverse highlighted when displayed on hardware controllers that support it. Not all hardware controllers utilize reverse highlighting.
- Next, orientation flags for each plug-in control will be displayed. The format is: "(Value: xxx yyy)," where xxx is either "BMin" (`kDAE_BottomMinTopMax`) or "TMin" (`kDAE_TopMinBottomMax`); and yyy is either "LMin" (`kDAE_LeftMinRightMax`) or "RMin" (`kDAE_RightMinLeftMax`). This text identifies the value of the control's orientation flags, as returned by `GetParameterOrientation()`.
- Also shown is the radial LED encoder-display mode (as returned by `GetControlOrientation()`) assigned to each control (this radial LED surrounds each encoder). The possible modes are: "spread" `kDAE_RotarySpreadMode`, "wrap" `kDAE_RotaryWrapMode`, "boost" `kDAE_RotaryBoostCutMode`, or "dot" `kDAE_RotarySingleDotMode`, along with either "RMin" `kDAE_RotaryRightMinLeftMax`, or "LMin" `kDAE_RotaryLeftMinRightMax` - indicating which side the minimum AAE value is being mapped to.

12.43.7.5 Control Highlighting Scheme

Note that plug-in controls that are currently controllable on any page of a plug-in will be highlighted in blue when they are the active page on a control surface. Therefore, it is very important to implement the highlighting of controls in your plug-in. Plug-in highlighting is also used with automation. In general, four common colors should be implemented:

- Red: Write automated
- Green: Read automated
- Blue: Accessible on control surface (stays blue if control is also read automated)
- Yellow: Accessible on control surface and write automated

You can use the "hidden" popup menu above to test that your color schemes are working properly.

12.43.7.6 Control Numbering Layouts

Most of the advanced control surfaces have both rotary encoders (knobs) and switches. The knobs and switches are handled differently by different surfaces. C|24 has a set of 24 rotary encoders and 24 switches for plug-in editing, and 003 has 8 encoders and 8 switches, all set up in pairs. However, both of these control surfaces automatically assign a control with only two possible values (e.g., "on" or "off") to a switch, while a control with three or more possible values, whether it is discrete or continuous, is automatically assigned to the rotary encoder. Therefore, no distinction is made between control numbers for switches and control numbers for encoders.

The following tables show how the control numbers are arranged for control surfaces which have distinctions between their encoders and switches.

Table 12.26 Table 2: D-Control Channel Strip - Numbering Layout

Encoders	Switches
1	7
2	8
3	9
4	10
5	11
6	12

Table 12.27 Table 3: D-Control/Pro-Control Custom Fader Mode - Numbering Layout

Encoders	Switches
1	9
2	10
3	11
4	12
5	13
6	14
7	15
8	16

12.43.7.7 Alphanumeric Displays

With the advent of the newer advanced control surfaces (CS), plug-ins now have the opportunity to provide information to the user via alphanumeric displays on the CS. Unfortunately, the displays are limited in the number of characters that can be shown. For instance, on the Mackie HUI, nine characters are provided for each plug-in control, allowing only four characters for the control name, and four characters for the control value, and one for a space between them. On ProControl, eight characters are provided for each plug-in control, with three characters allocated to displaying a control name, and four characters for its control value. On C|24 and 003, four characters are provided for the plug-in name, control name, and the control value. For the control value, these four characters include a +/- and/or any necessary unit abbreviations (ex: K, s, dB, etc.). D-Control and Command|8 provide six characters for the plug-in name, control name, and control value.

In order to display meaningful information in these short character strings, plug-ins will have to optimize the strings that they return to AAE. The dispatcher calls `MapControlValToString()`, which in turn calls `GetValueString()`, is used to obtain these control value strings. Typically in the past, the requested length argument of both these member functions, i.e., `maxChars` in `MapControlValToString()` or `maxLength` in `GetValueString()` has been ignored; but, from here on out, they should be carefully examined and used to create an optimum and meaningful string for any requested length.

To prevent the code from becoming unwieldy, as in the case of trying to provide strings for all requested lengths, a minimum number of expected lengths should be specifically addressed. The expected value lengths are: 4, 5, 6, 7, 8, and 31. In addition, ProControl switch states are a maximum length of 3 (i.e., when dealing with the state of a switch for ProControl, provide this information in 3 characters or less). Therefore, whenever possible, a plug-in should return the most meaningful string that will fit in any particular requested length, and at minimum, handle the expected lengths. If a plug-in does not have custom code to handle a particular requested length, it can round the length down to the next smaller expected length and use the code that it has for it. For example, a request of 9 characters should be converted to an expected request of 8 characters.

Please note: Since truncating a long string to fit within the requested length will not provide meaningful results in most cases, plug-ins must specifically provide code for deriving useful strings for the expected lengths where applicable.

Since the smaller lengths, especially 4 and 5 characters, are usually too short to display a plug-in's true full value including units, some decisions will have to be made about how to suitably shorten them. The following provides some general guidelines.

If needed, and in order of precedence, try to:

- Remove spaces: 13 Hz becomes 13Hz
- Use common abbreviations for units: 16 seconds to 16sec, or 16 s, 156 Hertz to 156Hz
- Drop the units entirely: 1832 Hz to 1832
- Round the value: 173.3 ms to 173

Here is a table depicting a typical example in more detail. The expected lengths are shown in the left most vertical column.

Alphanumeric Characters

While creating the above strings, the requested length argument passed in (`maxChars` in `MapControlValToString()`, and `maxLength` in `GetValueString()`) should be strictly adhered to! The string returned should be no greater than the requested number of characters. Furthermore, the developer should assume that the buffer passed into this function is only as large as the requested length. Any intermediate string processing should be done in temporary local buffers and only when you have the final string should you copy back to the buffer that was passed in, making sure you copy no more than the requested number of characters, plus the Null character or Length byte, as appropriate; since in general, `MapControlValToString()` uses C strings and `GetValueString()` uses Pascal strings.

Also, in an effort to further help prevent buffer overruns, two new functions have been added to the PI library file `SliderConversions.cp`: `SmartAppendNum()` and `SmartAppendXNum()`, which includes a maximum length argument and should be used in place of `FicAppendNum()` and `FicAppendXNum()` from `FicBasics.cpp`.

Finally, note that `ProControl` and `HUI` will sometimes utilize 5 characters to display its value when a sign is involved. For instance, if the number is -100, the negative sign will appear in the space separating the control name from the control value. `Pro Tools` will take care of this conversion as long as all of the expected lengths are properly provided for. For `C|24` and `003`, this is not the case - only four characters are allowed, so the +/- must be a part of those four characters.

`HUI`, `ProControl`, `C|24`, and `003` all provide alphanumeric displays for visual feedback, with the primary purpose being plug-in parameter editing. Functions in the plug-in library are provided that allow customized parameter strings to be created for use on the display. Specifically, these functions are: `GetControlNameOfLength()` and `GetValueString()`. Fortunately, the details of writing to the display are taken care of by the application. Therefore, the plug-in developer only needs to be concerned with providing meaningful display strings for all plug-in parameters that are controllable. Whether using the XML or legacy page table system, `GetControlNameOfLength()` should return the long version (31 characters maximum) of the plug-in's control names. Short versions of plug-in control names are stored in the XML file, edited with the [Page Table Editor](#) application. If you are also using legacy page tables to support versions of `Pro Tools` prior to 6.4, the short names should also be coded in the `GetControlNameOfLength()` function.

AAE clients like `Pro Tools` call `GetControlNameOfLength()`, but if AAE finds XML data stored in the plug-in, it gets the information from there rather than calling into the plug-in. `GetControlNameOfLength()` is responsible for providing the parameter "names" used on the display. As with parameter "value" strings, it is important to carefully create parameter names that are meaningful in the limited space allocated to displaying parameter names. On `ProControl`, string lengths of three characters will be used for the parameter name strings, where four characters will be used on the `HUI`, `C|24`, and `003`. `D-Control` and `Command|8` use six characters for control names. In general, lengths of 3, 4, 5, 6, 7, 8, and 31 should be specifically addressed in the `Page Table Editor` or `GetControlNameOfLength()`. We begin next, by looking at some concrete examples.

12.43.7.8 ProControl Display

`ProControl` also provides an alphanumeric display; however, there are some significant differences that need to be addressed. Shown is a generic representation for the `ProControl` display. The image below shows what users will usually be viewing on `ProControl`'s displays, which are the current Encoder/Value settings. Figure 13 shows the current switch settings when the user toggles to the Switch/State display mode. `ProControl` will only display one of three views at any given time.

Pro-Control Encoder Display

ProControl Switch Display

As with the `HUI`, meaningful strings will have to be provided in the limited available lengths. `ProControl`, `C|24` or `003` value strings require no special treatment other than what has already been stated above for `HUI`, since all

of these control surfaces display their values in 4 digits/characters, as returned by `GetValueString()`. The biggest difference between the HUI display and the Avid control surfaces' displays is that the Avid control surfaces can display symbols.

Let's take a moment to explain how the displays of C|24 and 003 work. Both of these surfaces have a four character LED display located above each encoder/switch pair. When in Channel mode, these scribble strips show the plug-in name (if any) on that channel. When that plug-in is selected, the display switches to show the controls for the plug-in. Now each display shows the name of the control. The display automatically switches to the current value of the control when the encoder or switch is moved or pushed.

12.43.8 Appendix A. Get Parameter Value Info

12.43.8.1 Overview

```
AAX_Result GetParameterValueInfo ( AAX_CParamID iParameterID, int32_t i↔
Selector, int32_t* oValue)
```

`GetParameterValueInfo()` is implemented at the Data Model's [AAX_CEffectParameters](#) level, and will allow the app to query a plug-in for the "meaning" of its parameter values. It was designed as a general purpose mechanism that will find additional uses with new selectors in the future. It is used:

- to ensure the EQ and Dynamics sections' EQ type selector LEDs light appropriately for a given band's filter type. We want the appropriate EQ type LED to light for each band's filter type -whether or not your band can switch between filter types.
- to ensure the state of the EQ section's In buttons matches the values of the associated plug-in controls. When each band's In button is lit, it must mean that the EQ is active/on/enabled. When it is unlit, it must mean that the EQ is off/disabled/bypassed. (Some plug-ins have EQbypass controls (On = bypass); others have EQ In buttons (On = On). We can derive "meaning" from the control regardless of control value.)
- similarly, to ensure the state of the Dynamics section's Filt In buttons matches the values of the associated plug-in controls. When each band's Filt In button is lit, it must mean that the EQ is active/on/enabled. When it is unlit, it must mean that the EQ is off/disabled/bypassed.

12.43.8.2 Implementation

`GetParameterValueInfo()` will be of type [AAX_EParameterValueInfoSelector](#) ([AAX_Enums.h](#)) and will allow for future queries on parameter value "meaning."

```
enum AAX_EParameterValueInfoSelector
{
    AAX_ePageTable_EQ_Band_Type = 0,
    AAX_ePageTable_EQ_InCircuitPolarity = 1,
    AAX_ePageTable_UseAlternateControl = 2
};
```

Listing 3: Parameter Selector Enums

Results (not return values) passed back by `GetParameterValueInfo()` will be of one of these two types:

```
enum AAX_EEQBandTypes
{
    AAX_eEQBandType_HighPass = 0,
    AAX_eEQBandType_LowShelf = 1,
    AAX_eEQBandType_Parametric = 2,
    AAX_eEQBandType_HighShelf = 3,
    AAX_eEQBandType_LowPass = 4,
    AAX_eEQBandType_Notch = 5
};
```

Listing 4: EQ Band Type Enums

```
enum AAX_EEQInCircuitPolarity
```

```
{
    AAX_eEQInCircuitPolarity_Enabled = 0,
    AAX_eEQInCircuitPolarity_Bypassed = 1,
    AAX_eEQInCircuitPolarity_Disabled = 2
};
```

Listing 5: EQ Circuit Polarity Enums

```
enum AAX_EUseAlternateControl
{
    AAX_eUseAlternateControl_No = 0,
    AAX_eUseAlternateControl_Yes = 1
};
```

Listing 6: Alternate Control Enum

Please see [AAX_Enums.h](#) for more information.

To add support for this method, you must to override [AAX_CEffectParameters::GetParameterValueInfo\(\)](#) from within your plug-ins Data Model. For a given [AAX_CParamID](#), selector, and parameter value, you must pass back a result (not a return value) denoting the "meaning" of that parameter value. If a parameter has no meaning in the context of the given selector, you should return [AAX_ERROR_UNIMPLEMENTED](#).

One point to note, is that an EQ or Dynamics plug-in may have a band of EQ that does not include an EQ type selector control. The band is just always, say, a HPF. There's no control to map to the EQ type selector button in the page table and therefore no obvious control for which you would pass back [AAX_eEQBandType_HighPass](#) in [GetParameterValueInfo\(\)](#). What is the solution? Well, the other controls on that band (Frequency, Q/Slope, Gain, In) know what type of band they're on. Thus the plug-in should pass back the relevant [EQEQ_Band_Types](#) enum in [GetParameterValueInfo\(\)](#) for all controls on a given band of EQ. This also applies to key filter bands in your Dynamics plug-ins. In this way, the EQ type selector LEDs in both the EQ and Dynamics sections will always light appropriately.

The second exception applies to EQ or Dynamics plug-ins that have a band of EQ that does not include an In Circuit / Out of Circuit control for that band. In this case, the band is always In Circuit (or On) and the other controls for this band should return [AAX_eEQInCircuitPolarity_Enabled](#) as the result for [GetParameterValueInfo\(\)](#) when the [AAX_ePageTable_EQ_InCircuitPolarity](#) selector is passed in. Code snippets for [GetParameterValueInfo\(\)](#) from the EQ III 7-Band AAX plug-in is provided below:

Note

The logic for the In Circuit / Out of Circuit LED is available in Pro Tools 6.7 and higher.

```
// *****
// METHOD: GetParameterValueInfo
// *****
AAX_Result EQIII_7_Parameters::GetParameterValueInfo ( AAX_CParamID iParameterID, int32_t iSelector,
int32_t* oValue) const
{
    const AAX_IParameter * parameter = mParameterManager.GetParameterByID( iParameterID );

    if ( !parameter )
        return AAX_ERROR_INVALID_PARAMETER_ID;

    if ( iSelector == AAX_ePageTable_EQ_Band_Type )
    {
        if ( parameter->Name() == EQIII_HPF_Type )
        {
            switch( parameter->GetStepValue() )
            {
                case 0 /*Notch*/: *oValue = AAX_eEQBandType_Notch; break;
                case 1 /*HiPass*/: *oValue = AAX_eEQBandType_HighPass; break;
                default: *oValue = -1; return AAX_ERROR_UNIMPLEMENTED; break;
            }
            return AAX_SUCCESS;
        }
        else if ( parameter->Name() == EQIII_LF_Type )
        {
            switch( parameter->GetStepValue() )
            {
                case 0 /*Peak*/: *oValue = AAX_eEQBandType_Parametric; break;
                case 1 /*Shelf*/: *oValue = AAX_eEQBandType_LowShelf; break;
                default: *oValue = -1; return AAX_ERROR_UNIMPLEMENTED; break;
            }
        }
    }
}
```

```

        return AAX_SUCCESS;
    }
    else if (parameter->Name() == EQIII_HF_Type )
    {
        switch (parameter->GetStepValue() )
        {
            case 0 /*Peak*/:  *oValue = AAX_eEQBandType_Parametric; break;
            case 1 /*Shelf*/:  *oValue = AAX_eEQBandType_HighShelf; break;
            default: *oValue = -1; return AAX_ERROR_UNIMPLEMENTED; break;
        }
        return AAX_SUCCESS;
    }
    else if ( parameter->Name() == EQIII_LPF_Type )
    {
        switch ( parameter->GetStepValue() )
        {
            case 0 /*Notch*/:  *oValue = AAX_eEQBandType_Notch; break;
            case 1 /*LoPass*/:  *oValue = AAX_eEQBandType_LowPass; break;
            default: *oValue = -1; return AAX_ERROR_UNIMPLEMENTED; break;
        }
        return AAX_SUCCESS;
    }
}
else if (iSelector == AAX_ePageTable_UseAlternateControl)
{
    if ( (parameter->Name() == EQIII_HPF_Type ) ||
        (parameter->Name() == EQIII_HPF_Q) ||
        (parameter->Name() == EQIII_HPF_Slope) )
    {
        const AAX_IParameter* typeParam = mParameterManager.GetParameterByID(EQIII_HPF_Type_Ch);
        *oValue = (typeParam->GetStepValue() == 0 /*Notch*/) ? AAX_eUseAlternateControl_No :
AAX_eUseAlternateControl_Yes;
        return AAX_SUCCESS;
    }
    else if ( (parameter->Name() == EQIII_LPF_Type) ||
        (parameter->Name() == EQIII_LPF_Q) ||
        (parameter->Name() == EQIII_LPF_Slope) )
    {
        const AAX_IParameter* typeParam = mParameterManager.GetParameterByID(EQIII_LPF_Type_Ch);
        *oValue = (typeParam->GetStepValue() == 0 /*Notch*/) ? AAX_eUseAlternateControl_No :
AAX_eUseAlternateControl_Yes;
        return AAX_SUCCESS;
    }
}
return AAX_ERROR_UNIMPLEMENTED;
};

```

Listing 7: EQIII GetParameterValueInfo()

As you'll notice, the EQ III plug-in has separate controls for Q and Slope in its HPF and LPF bands, depending on whether the band is set to notch or band pass. When the Band Type control is set to notch, the continuous Q control is used. When the Band Type is set to Hi/Lo Pass, the discrete Slope control is used. In the page table, the HPF / LPF Q controls are set to the "Q or Slope" in the [Page Table Editor](#) application, and the HPF / LPF Slope controls are set to the "Q or Slope Alt". Then, with the [GetParameterValueInfo\(\)](#) implementation above, the control surface properly swaps the controls when the band type control is changed. In other words, when the function is called with the [AAX_ePageTable_UseAlternateControl](#) selector, if the band type control is set to notch, then [AAX_eUseAlternateControl_No](#) is returned in the result and the control surface puts the Q control at that position. If the band type is set to pass, then [AAX_eUseAlternateControl_Yes](#) is returned and the Slope is placed at that position. Collaboration diagram for Page Table Guide:

AAX Host Guides

Page Table Guide



12.44 DigiTrace Guide

How to add tracing to your plug-ins and view logging from the plug-in host.

12.44.1 On this page

- [What is DigiTrace?](#)
- [DigiTrace quick start guide](#)
- [DigiTrace log files](#)
- [Configuring DigiTrace](#)
- [Bonus features](#)
- [Adding traces to an AAX plug-in](#)
- [Advanced DigiTrace configuration](#)
- [Compatibility](#)
- [Additional Information](#)

12.44.2 What is DigiTrace?

DigiTrace is a logging tool used by many Avid audio applications. DigiTrace provides high-performance, real-time tracing capabilities and can help you debug hard-to-isolate problems in real-time code. Pro Tools and other Avid audio products are instrumented with DigiTrace, and it is easy to add DigiTrace logging to your AAX plug-ins.

This document outlines how to use DigiTrace, both as a developer to add trace instrumentation to your code and as an end user to view or record trace instrumentation for an instrumented application.

12.44.2.1 What does DigiTrace do?

DigiTrace generates encrypted logs on users' systems. These log files can be decrypted via the DigiTraceDecryptor application that is included in the DigiTrace Tools package.

By default, DigiTrace logs basic information including details about the system, software, component versions, and any errors that are encountered. By using a simple configuration text file, DigiTrace can be easily configured to provide additional logging information such as plug-in loading details. Here are some examples of how you can use DigiTrace:

- You can use DigiTrace in your plug-ins when you need a convenient, high-performance logging solution.
- You can use the default DigiTrace logs that Pro Tools generates to help you understand problems that your plug-ins encounter when running on Pro Tools.
- You can add DigiTrace statements and stack traces to your released plug-ins in order to help you troubleshoot end-user issues more quickly.
- You can (and should!) submit DigiTrace logs when reporting bugs and other Pro Tools issues to Avid.

12.44.3 DigiTrace quick start guide

This section provides quick steps for the following common tasks:

- [Find and decrypt DigiTrace log files](#)
- [Configure DigiTrace for AAX plug-in logging](#)
- [Configure DigiTrace for plain-text output](#)
- [Add tracing to a plug-in](#)

12.44.3.1 Find and decrypt DigiTrace log files

DigiTrace log files are placed into a common logs directory. The specific directory that is used depends on the version of DigiTrace - see [Where are DigiTrace log files stored?](#)

By default, the version of DigiTrace that is installed with Avid audio products generates logs in an encrypted format with the extension ".dlog". Developer builds of Pro Tools and other applications are configured to generate plain-text logs.

You can convert .dlog files to plain-text using the DigiTraceDecryptor tool that is included in the DigiTrace Tools package available for download from the Avid developer portal. To decrypt a log using this tool, simply drag-and-drop the .dlog file onto the tool. You can also set this tool as the default application for opening .dlog files in your OS, which will allow you to decrypt and open .dlog files directly.

12.44.3.2 Configure DigiTrace for AAX plug-in logging

You must customize the DigiTrace configuration to enable extra logging, such as debug logging for [AAX](#) plug-ins.

DigiTrace uses a plain-text configuration file to enable custom logging. This file uses the suffix ".digitrace" and is located within or beside the application. For example, the configuration files for Pro Tools are located at:

- macOS: Pro Tools.app/Contents/Resources/config.digitrace
- Windows: C:\Program Files\Avid\Pro Tools\ProTools.digitrace

To configure DigiTrace to print logs from [AAX_TRACE](#) or [AAX_TRACE_RELEASE](#) macros in [AAX](#) plug-ins, add the following line to the .digitrace configuration file for the application:

```
DTF_AAXPLUGINS=file@DTP_LOWEST
```

If a config.digitrace file does not already exist for a DigiTrace-enabled application then you can create it to enable DigiTrace. For more information about customizing the DigiTrace configuration and enabling different levels of debug logging, see [Configuring DigiTrace](#).

12.44.3.3 Configure DigiTrace for plain-text output

In order to be able to view streaming log output in real time, DigiTrace must be configured for plain-text output. This is the default configuration for developer builds of Pro Tools and other Avid audio applications.

To configure shipping applications for plain-text log output, you must replace the application's installed DigiTrace library with a development version of the DigiTrace library. Development builds of DigiTrace are included in the DigiTrace Tools package. Search for "DigiTrace.framework" on macOS or "DigiTrace.dll" on Windows and replace the installed shipping version of the library with the developer version from the DigiTrace Tools package to configure the application for plain-text output.

Note that the developer version of DigiTrace may output logs to a different directory than the shipping version. In general, developer builds of DigiTrace will place log files in a directory next to the instrumented application. For example, developer builds of Pro Tools will output logs to a logs directory placed adjacent to the Pro Tools application bundle rather than in the user's Library/Logs/Avid folder.

12.44.3.4 Add tracing to a plug-in

To easily add tracing to an [AAX](#) plug-in, use the [AAX_TRACE](#) or [AAX_TRACE_RELEASE](#) macros. Logging from the "release" macro will be enabled for all builds of the plug-in, whereas logging from the "standard" macro will only be enabled in Debug builds of the plug-in.

12.44.4 DigiTrace log files

The default logging in Avid audio applications includes data that can be useful in many different troubleshooting situations. For example:

- Information about the user's system configuration
- A complete list of loaded components and libraries. (If the user has an old or incompatible version of your plug-ins installed on his system, you will know about it!)
- Crash logs in the event of a system failure

In addition, you can add DigiTrace logging code to your plug-ins, helping you examine potential issues in the way your plug-in is running on a user's computer even when you cannot reproduce the issue locally.

12.44.4.1 Where are DigiTrace log files stored?

12.44.4.1.1 Log directory DigiTrace logs are stored in a log files directory on the user's system:

~/Library/Logs/Avid/ (macOS) %userprofile%\AppData\Local\Avid\Logs or C:\Program Files\Avid\Pro Tools\Logs (Windows)

This default log directory can be overridden in the DigiTrace config file. See [Advanced DigiTrace configuration](#) for more information.

12.44.4.1.2 Log file names By default the log file will be given a time-stamped name in the format <App↵Name>_YYYY_MM_DD_HH_MM_SS.dlog. This timestamp represents the system time when the log was created. Like the log directory, this log file name can be changed using the DigiTrace configuration. See [Advanced DigiTrace configuration](#) for more information.

12.44.4.2 Monitoring DigiTrace logs

12.44.4.2.1 Log files You can of course view a log file by opening it periodically. In addition, assuming that DigiTrace is [configured for plain text output](#), you can also constantly monitor a log file in a "streaming" manner. This is possible using standard Unix tools included with macOS or with Cygwin on Windows. In fact, this approach usually works better than telling DigiTrace to use console output due to buffering of the console output.

- For basic real-time monitoring of a single file, use `tail: tail -f /path/to/digitrace/logs/the_logfile.txt`
- For real-time monitoring of the most recent file in the log file directory, use a combination of `tail` and `ls`:

12.44.4.2.2 Console

Console behavior is quite different between macOS and Windows

Windows On Windows, traces sent to the console go to the system debugging console. The only way to view the console output is to be running with an attached debugger.

macOS On the Mac, console traces are sent to stdout console, which shows up in a few places:

- If you're running in a debugger, the debug console will display stdout output, including DigiTrace messages
- If you're not in the debugger, you can view the output in the Console app (/Applications/Utilities/Console). For Pro Tools, look under ~/Library/Logs/Avid/Pro Tools.X.log in the log list. Note that these messages are not displayed in the "All Messages" log.
- Alternately, you can manually look at the log output, again using the `tail` command, e.g. `tail -f "~/Library/Logs/Avid/Pro Tools.0.log"`

12.44.4.3 Log file formatting

Here is the beginning of an example DigiTrace log:

```
*** Digidesign Session Trace for: /Applications/Pro Tools 11.0.2 3PDev.app (pid=0x5aff, version=11.0.2d626)
*** Starting Timestamp: Tuesday, January 7, 2014 4:10:57 PM Eastern Standard Time (89706938666 uS)
*** System Details: OS Version: 10.8.5, CPU Speed: 2.7 GHz, Architecture: Intel 64 bit, Num Processors: 8
*** DigiTrace Config File: /Applications/Pro Tools 11.0.2 3PDev.app/Contents/Resources/config.digitrace
*** Facilities to trace:
```

```
DTF_INSTALLED_COMPONENTS@DTP_NORMAL(0e0d)
```

```
Time(us),Tid,Facility,Name : Debug Message
```

```
-----
89707181683,00c07,0e0d: Pace eden lib version: 2.0.0, r22343 (2.0.0.22343), [...]
89707220338,00c07,0e0d: ShoeTool_Init - shoe tool installed version is 6.000 [...]
89707220374,00c07,0e0d: ShoeTool_IncreaseAIOLimits - var=46, newVal=512, cur [...]
89707220380,00c07,0e0d: ShoeTool_IncreaseAIOLimits - var=47, newVal=512, cur [...]
```

The log file consists of a header followed by a series of log statements. Each log statement includes the following information:

- Time(us) - The time the message was logged, in microseconds since the machine was started.
- Tid - The thread ID of the thread that logged the message.
- Facility - The Facility ID of the facility that's logging the message.
- Name - This is the config name added to all facilities included by this config file. This can be used to group all facilities related to a feature set, for instance. If not set, this is not included.
- Debug Message - This is the actual string passed to the trace facility.

12.44.5 Configuring DigiTrace

You can configure DigiTrace to include or exclude specific traces using the `config.digitrace` configuration file. This file is plain text and includes a single configuration command on each line.

This is the basic format for a command used to enable tracing for a single [facility](#):

facility=[console@minimum console logging priority],[file@minimum file logging priority]

Here are some examples:

- `DTF_APP_VERSION=file@DTP_LOW`
- `DTF_PLUGINS_3P=file@DTP_LOW,console@DTP_URGENT`
- `DTF_ASSERTHANDLER=console@DTP_URGENT`
- `DTF_DAE_MEM=console@DTP_URGENT,file@DTP_LOWEST`

For more information about special configuration commands, see [Advanced DigiTrace configuration](#).

12.44.5.1 Trace facilities

Trace facilities are used by DigiTrace to determine whether or not the given trace statement should be displayed. Trace facilities allow the user to filter trace statements at the component level.

12.44.5.2 Trace priorities

Trace priorities are used by DigiTrace to determine whether or not the given trace statement should be displayed. DigiTrace specifies five trace priorities:

- `DTP_LOWEST`
- `DTP_LOW`
- `DTP_NORMAL`
- `DTP_HIGH`
- `DTP_URGENT`

The DigiTrace configuration file specifies minimum trace priorities. For example, if a trace statement uses `DTP_LOW` and DigiTrace is configured to use `DTP_NORMAL` as the minimum trace priority, then the trace statement will not be sent to the output target. In general, the `DTP_LOWEST` priority setting will populate the trace output with the most verbose information while the `DTP_URGENT` setting will output only the most high level details.

12.44.5.3 Useful DigiTrace facilities

This section includes descriptions of several facilities that are used in Pro Tools and other Avid audio products. The logging provided by these facilities may be helpful when diagnosing plug-in issues.

- **DTF_AAXPLUGINS** This is the standard facility for **AAX** plug-ins. This facility will only log traces that are present in **AAX** plug-ins themselves, not traces in any hosting code. Plug-ins may use the **AAX_TRACE** or **AAX_TRACE_RELEASE** macros to log to this facility.

Note

Disabling the **DTF_AAXPLUGINS** facility will slightly reduce the overhead of trace statements and chip communication on HDX systems.

- **DTF_AAXHOST** at **DTP_NORMAL** Logging from the main **AAX** host component. Use a lower priority for additional **AAX** Host tracing.
- **DTF_PLUGINS** at **DTP_LOW** Miscellaneous plug-in operations, including page table logging, preset directory errors, and DLL loading and unloading
- **DTF_TIPLUGINS** at **DTP_NORMAL** Logging for HDX plug-in algorithm handling details such as packet management and private data field state reset. Use **DTP_LOW** for deeper tracing.
- **DTF_TISHELLMGR** at **DTP_HIGH** Logging from the HDX RTOS
- **DTF_DAE_HOSTDEVICE** at **DTP_URGENT** Performance logging from the real-time audio render thread. See [Real-time AAE performance logging with DigiTrace](#)
- **DTF_DAE_ERRORS** at **DTP_NORMAL** or **DTP_LOW** Information about any errors that occur in AAE. Use **DTP_LOW** to enable stack traces.
- **DTF_ASSERTHANDLER** at **DTP_NORMAL** or **DTP_LOW** Similar to **DTF_DAE_ERRORS**: Information about any asserts that fail. Use **DTP_LOW** to enable stack traces.
- **DTF_THREAD_NAMES_AND_PRIORITIES** at **DTP_NORMAL** or **DTP_LOW** Allows you to look up a thread's debug name from its ID on the standard trace line. Thread names and IDs will be traced as they are created, and you can then use that ID to resolve the thread name of later trace statements. Use **DTP_↔NORMAL** for just names and IDs, and **DTP_LOW** to include priorities.
- **DTF_PACESUPPORT** at **DTP_NORMAL** Plug-in digital signature logging, with some diagnostics for digital signature verification failures.
- **DTF_ADC** at **DTP_NORMAL** Delay compensation logging, including host accounting for plug-in latency.
- **DTF_AUTOMATION** at **DTP_LOW** Parameter touch and release logging.
- **DTF_AUDIOSUITE** at Logging of events specific to AudioSuite plug-in instances.

12.44.6 Bonus features

12.44.6.1 Real-time AAE performance logging with DigiTrace

Pro Tools 11 and higher includes logging for audio render callback performance. To enable this logging, enabled the **DTF_DAE_HOSTDEVICE** facility at **DTP_URGENT**. This facility will enable logging of real-time audio render thread metrics around any render errors that occur.

Here is an example of a performance log:

```
Int(LL): hstEr=0, ioEr=0, dif=2665(2891,23129), tot=2517(1648,2317), in=51(58,83),
clbk=2158(1508,2158), out=108(99,164), offset=[12], mxW=1101(com.avid.aax.↵
eleven.free)
```

The different values included in this log are:

- `hstEr`
- `ioEr`
- `dif`
- `tot`
- `in`
- `clbk`
- `out`
- `offset`

A log of 'x=a (b,c)' means that the (x) value (e.g. `tot`) for the interrupt was (a) us, the running average was (b) us, and the maximum value encountered was (c) us. Therefore, in the example above:

- 1648 us average total time was spent in each interrupt
- 2517 us was spent in this interrupt
- Eleven Free was the longest worker in this interrupt

In practice, it is difficult to precisely log this information during an error. This is due to changes in the interrupt pattern and scheduling when the audio engine is halted. In order to account for this, the performance logging will print out logs for several interrupts around when any error occurs. The actual audio engine error (`hstEr`) may be reported for a "junk" interrupt cycle that is spuriously logged during this halt process.

12.44.6.2 Adding signposts to the DigiTrace log at run-time

Use the shift-`~` key combination to add a "Trace Flag" line into the DigiTrace log. This allows you to add a "signpost" line to the log right when an important event happens, or before/after an important operation, so that it is easier to find the important details when inspecting the log later.

```
176603608088,2b603,0016: DSK_PrePrimeDiskTask::PrePrimeDiskTask - finish
176603961348,00307,0000: Trace Flag 3 (diff prev: 2.40s, diff start: 7.18s)
176605252296,00307,0000: Trace Flag 4 (diff prev: 1.29s, diff start: 8.47s)
176605252779,00307,0f09: 2016-07-19 23:36:32.499 PTC_Mgr::Idle() -- performing task (Websocket Base)
176606039830,00307,0000: Trace Flag 5 (diff prev: 0.79s, diff start: 9.26s)
```

Each trace flag signpost includes the text "Trace Flag" and the diff (in seconds) from both the previous trace flag and the first trace flag which was triggered during the current run of the app.

These lines will be printed regardless of the current DigiTrace configuration settings.

Host Compatibility Notes This feature is available in Pro Tools 12.6 and higher

12.44.7 Adding traces to an AAX plug-in

12.44.7.1 Basic AAX logging

Standard `printf`-style logging from AAX plug-ins is very easy. This feature is built into the AAX specification and is exposed to plug-ins via the `AAX_TRACE` and `AAX_TRACE_RELEASE` macros. For more information about basic logging, see the documentation for those macros.

Note

To enable basic AAX logging via these macros, the `DTF_AAXPLUGINS` trace facility must be enabled.

12.44.7.1.1 Tracing for AAX DSP The [AAX_TRACE](#) and [AAX_TRACE_RELEASE](#) macros, as well as [AAX_ASSERT](#), are cross-platform and are supported for use in AAX DSP algorithms. For more information about tracing from AAX DSP algorithms, see the [Tracing](#) section in the [HDX DSP Guide](#).

12.44.7.2 Advanced DigiTrace logging features

As a developer, you can use several advanced macros to extend the functionality of DigiTrace logging in your plug-in beyond the simple `printf`-style features provided by [AAX_TRACE](#). The full DigiTrace macro suite includes macros for stack traces, very long traces, or even the ability to dump a block of memory to the log.

Note that these advanced features are only available on the host system. They are not currently available from algorithms running on embedded hardware.

12.44.7.2.1 What files do I include in my project? To add advanced DigiTrace instrumentation to your source code you must:

1. Include `DigiTrace.h` in the file where you are going to put your trace statements.
2. Compile `CDigitraceAccess.cpp` into your project

If you have problems including the DigiTrace header file, try moving it to the top of the file that you're including it into. DigiTrace has no other dependencies and should be safe to include into any component.

12.44.7.2.2 What do I do if I encounter problems compiling or linking? Should you run into linker errors or other problems after adding the DigiTrace header file, please go through the following items and verify that each is included in your project:

- The `CDigitraceAccess.cpp` file automatically searches for and loads the `DigiTrace.dll` (Windows) or `DigiTrace.framework` (Mac) component and ensures that the appropriate function pointers are initialized. If you receive linker errors, the missing symbols are likely in this file. Note that your project will need the path to `CDigiTraceAccess.h` in order to compile this file.
- If you receive an include file error for `DigiPragmas.h` then you will need to add the header's path to your project's search paths. This file is included in the most recent DigiTrace Tools packages but may not be included in some older packages.

12.44.7.2.3 Macros DigiTrace provides five core macros for trace output, which are:

- `TRACE_R` - for general `printf` style tracing. Subject to a total line limit of 256 chars.
- `TRACE_PUTS_R` - prints an arbitrary length buffer, splitting it up into clean lines based on line breaks. No formatting.
- `STACKTRACE_R` - for stack trace printing. See below.
- `MEMTRACE_R` - for memory buffer hex tracing
- `FXTRACE_R` - for automatic function entry and exit tracing

12.44.7.2.4 Debug vs. release macros All of the macros listed above are general-use macros, which generate output in both Debug and Release builds. They each have a debug-only variant which excludes the trailing "<↔TT>_R</TT>". Like [AAX_TRACE](#), these debug-only versions compile to a noop in release builds.

The use of debug-only macros is not usually necessary due to the fact that release traces are encrypted and hidden from end users (but not from other developers.) As a best practice, we recommend using the "_R" version of a macro whenever possible. The debug versions of the macros should only be necessary in special circumstances where you specifically do not want to compile the code into release builds.

For more information about tracing in release builds see [Security concerns](#)

12.44.7.2.5 Syntax Adding a DigiTrace statement to your code is as easy as making a single function call thanks to DigiTrace's predefined macros. The basic macro syntax is:

```
MACRO_NAME( TRACE_FACILITY_NAME [ | TRACE_PRIORITY_LEVEL ], MESSAGE_STRING )
```

Here is a code sample:

```
bool my_function(char* data_buffer, int data_buffer_len)
{
    FTRACE_R( DTF_PLUGINS_3P, "my_function" ); // Automatically trace function entry and exit.
    // You need only to specify a trace facility.

    OSErr err = FrobnicateBuffer(data_buffer, data_buffer_len);
    if( noErr != err )
    {
        TRACE_R( DTF_PLUGINS_3P | DTF_HIGH, "Couldn't frobnicate the buffer: %s", OSErrToString(err) );
    }
    else
    {
        int cBytesToTrace = 64;
        MEMTRACE( DTF_PLUGINS_3P | DTF_LOW, "Data after frobnication", data_buffer, cBytesToTrace );
    }
}
```

12.44.7.2.6 Generating stack traces The `STACKTRACE_R` macro is very useful for getting stack traces of important events in the code like throwing errors (which can be thrown from many locations). One particularly useful feature of this macro is that it allows you to specify a facility and priority for the printf part of the stacktrace, e.g. `DTF_NORMAL`, and another one for the stacktrace step, e.g. `DTF_LOW`. See `DigiTrace.h` for a full list of macros and their documentation.

12.44.7.2.7 Turning off tracing in a specific file You can explicitly disable tracing in an instrumented file in the build by defining the `MTurnDbgTraceOff` symbol at the top of the file.

12.44.7.3 Security concerns

Unless you provide your own logging encryption, DigiTrace logs are not secure and should not be used to store any sensitive information.

Logs generated by Pro Tools release builds on users' systems are encrypted. This is primarily for the sake of avoiding confusion in our user community, since DigiTrace logs can be cryptic and potentially misleading for users who are not familiar with our code.

Avid and other third-party developers will see your plug-ins' release trace statements if they load your plug-ins with the appropriate trace facilities enabled. We highly recommend that you keep this in mind when developing your trace statements, both in order to prevent confusion (see the formatting guidelines in the [AAX_TRACE_RELEASE](#) documentation) and in order to maintain the security of your code.

12.44.8 Advanced DigiTrace configuration

The basic configuration command to enable tracing for a facility is described above in [Configuring DigiTrace](#).

There are also additional commands that can be added to the DigiTrace configuration file for more advanced configurations.

12.44.8.1 Configuration command format

- All DigiTrace configuration commands are listed in the configuration file with the form `<token>=<value>`
- Any blank line or line beginning with a '#' character is ignored
- Tokens are not case sensitive
- If there are repeated tokens in a file, the last token wins

12.44.8.2 Advanced configuration commands

- `FileTracingDir = { DIRPATH }`
 - Default: `USE_RELATIVE_PATH`
 - Custom log file directory. e.g. "C:\MyTraceDir" on Windows or "~/MyTraceDir" on macOS.
 - If `DIRPATH == USE_RELATIVE_PATH` then the output trace file directory will be created next to the target application.
- `Append = { true | false }`
 - Default: `false`
 - Append to file. If `true`, the output of this trace will be appended to any existing log file with the same name. Otherwise, this trace will overwrite an existing log file with the same name.
- `LogFileLimit = { LIMIT }`
 - Default: no limit
 - Limits the number of log files kept around for this config file to the specified number.
 - If set to an integer value, DigiTrace will delete the oldest log file(s) until there are only N most recent log files in the output folder.
 - If you rename an output file so it does not have the standard prefix, it is never deleted by this option.
 - Does not work with the "append" option
- `TraceQueueSize = { small | medium | large }`
 - Default: `small`
 - This controls the amount of memory allocated to the trace queue. You probably won't need to change it.
- `BeQuiet = { true | false }`
 - Default: `false`
 - "Quiet" mode. If set to `true`, this configuration option makes all trace output occur without any decoration (i.e. no timestamps, no thread id, no process id, etc.).
 - This mode may be useful for some types of real-time vector tracing or for configuring formatted logs for post-processing with a text editor.
- `FileId = { FILEID }`

- Default: none
- If set, this string is included in the filename created by DigiTrace for trace output files.
- Does not work with the "append" option
- Name = { NAME }
 - Default: none
 - If set, this string is included in every trace for all the facilities that are enabled in this config file.
 - You can have one of these per config file.

12.44.8.3 Dynamically changing the DigiTrace configuration

DigiTrace config files can be loaded dynamically, which means that you can add new configs while the instrumented application is running. Below are the details surrounding dynamic loading:

- In debug builds, this will happen each time the app comes to the foreground.
- The API only does anything if something has changed in your config files that will result in different tracing of some sort. If nothing has changed, the overhead to make the call is $< 1\text{ms}$, and current tracing is not affected.
- If something has changed, the changes are merged in to the existing config objects in memory. Active log files are not interrupted when possible. This takes around 200ms (mostly because of a sleep command that lets threads finish tracing things). Traces that happen in threads during this changeover will be dropped.
- No code in the actual tracing commands was changed.
- You can change any of the attributes of the trace facilities (priority level, file or console, etc).
- You can add new config files on the fly, or if you start with no config file, you can add one on the fly.

12.44.9 Compatibility

DigiTrace is an internal testing and troubleshooting tool. Although we will try to provide up-to-date documentation so that third-party developers can use this tool, we may need to change the way that DigiTrace works at some point and so we cannot make any promises regarding forwards-compatibility.

At the time of this writing:

- DigiTrace is fully compatible with Pro Tools 8.0.3 and later. DigiTrace is installed by default with compatible Pro Tools shipping versions and with some Pro Tools Development Builds.
- DigiTrace is compatible with all versions of the DAE dish in the DSH environment. Tracing must be explicitly enabled in the dsh executable by placing a config.digitrace config file next to the executable.

If you notice significant bugs or other problems with DigiTrace in any Pro Tools release then we encourage you to report the issues to us on the developer forum. We may not be able to address issues immediately, but your feedback is appreciated.

12.44.10 Additional Information

12.44.10.1 Confidentiality

As with all information provided in the [AAX SDK](#), the information provided in this documentation is confidential and is bound by the terms of your NDA with Avid. You may provide customized DigiTrace configuration files to end users in order to generate useful debugging information on their systems. However, you may not provide users with decrypted DigiTrace logs or with other details provided in this DigiTrace documentation.

Collaboration diagram for DigiTrace Guide:



12.45 DSH Guide

How to test basic functionality of AAX plug-ins using DSH test tool.

12.45.1 Contents

- [What is DSH and how it works](#)
- [Basic set of commands of the DAE dish](#)
- [Basic plug-in tests](#)
- [Debugging and tracing in DSH](#)
- [Scripting interface and batch profiling](#)

12.45.2 What is DSH and how it works

DigiShell is a software tool that provides a general framework for running tests on Avid audio hardware. As a command-line application, DigiShell may be driven as part of a standard, automated test suite for maximum test coverage. DSH supports loading all types of AAX plug-ins except AS, and is especially useful when running performance and cancellation tests of AAX-TI types. DigiShell is included in Pro Tools Development Builds as dsh.exe (Windows) or as dsh in the CommandLineTools directory (Mac).

After it is launched, DigiShell waits for a command name and parameters to be entered via stdin; command results are output via stdout. DigiShell parses its input as command name, followed by a single space, and then command parameters. The command parameters are expected to be a yaml-encoded string. Here are two examples of strings in compact (single-line) yaml format:

- A hash containing lists in compact yaml syntax `{ key1: [val1, val2], key2: [val3, val4] }`
- A list of two lists `[[PIO, 0, 1], [DSP, 1, 1]]`

12.45.3 Basic set of commands of the DAE dish

DigiShell has built-in commands for getting help, creating a DigiTrace configuration and loading DigiShell modules known as "dishes". One can see the command list by running the help command without any parameters. Passing a command name as a single string parameter to the help command will give a more detailed command description.

The default installation of DigiShell includes a few dishes, including the DAE dish. This dish loads the AAE audio engine and can be used for loading and testing basic functionality of plug-ins. The DAE dish can also be used to load a plug-in for basic debugging purposes, and provides a lighter-weight debugging environment than the full Pro Tools application.

Another dish supplied with DSH, the aaxh dish, provides a lower-level hosting environment for [AAX](#) plug-ins. This dish loads a dedicated [AAX](#) host component without audio routing logic or other audio engine responsibilities. In this guide we will focus on loading and running plug-ins using the DAE dish, but we encourage you to also explore the commands available in the aaxh dish and to learn how to exercise your plug-ins in that environment.

12.45.3.1 Loading plug-ins in DSH

The following commands can be used to load and configure the DAE dish:

- `load_dish DAE` Loads the DAE dish
- `init_dae 48000` This command is optional. It configures AAE to work at a specific sample rate. By default it will work at 44100 Hz.

Loading the DAE dish into the DigiShell environment with the built-in `load_dish` command will extend the set of available commands. Among them there is a `run` command. The `run` command can be used in two ways:

- Execute with no arguments: List all plug-in configurations which are available to AAE
- Execute with arguments specifying a particular plug-in configuration: Load a plug-in instance using the specified configuration

You can also search for the id and spec of the specific plug-in with the `findpi` command, which takes a plug-in's name or part of it as an argument, and then searches through the whole list of available plug-ins using this pattern.

Figure 1: DSH command for loading plug-ins.

If the plug-ins was instantiated successfully, then DSH will list all its parameters, just like on the screenshot. If instantiation fails, then DSH in most cases will output the error code, although it is not always obvious what this error code means. Here is the list of possible reasons of some failures:

- -9060 failed to load DSP Hybrid plug-in
- -14140 IO interface is not connected
- -7050 not enough resources for instantiating plugin
- -14378 plug-in exceeded memory limits
- -14003 something is wrong with your HDX card

-30xxx errors are dynamically-generated and can indicate different failures. Failures due to plug-ins exceeding the cycle limit of the DSP CPU will often appear as -30xxx errors. See [-30xxx: Dynamically-generated error codes](#) in the [HDX DSP Guide](#) for more information.

Note

`run` command works for Native and DSP plug-ins, but not for the Audio Suite ones. Also it will fail for DSP Hybrid plug-ins. To be able to instantiate them, one should run `acquiredeck` command.

There are several DAE dish commands for operating with plug-ins' instances:

- `getcurrentinstance` Returns the index of the current instance. The counting starts from 0 for the first instance that has been instantiated, and increments by one for every next instance.
- `getinstanceproperties` Returns the effect name for the Native plug-ins, and much more detailed info for the DSP instances.
- `setcurrentinstance` Sets the instance with the given index as the current instance.

12.45.3.2 Working with HDX card from DSH

One of the benefits of the DAE dish in DigiShell is that it has direct access to the shell environment that loads DSP plug-ins. The included facilities for retrieving load error information from the DSP manager can be very helpful for debugging DSP plug-in load failures. For example, you can use the following DAE dish commands to determine what resource requirement is preventing additional instances from loading onto a single DSP:

1. `reservetidsp all` Reserves all unused DSPs in the system
 2. `unreservetidsp 0` Frees the first DSP for plug-in allocation
 3. `getlastdsploadererror` Retrieve the text of the error that was generated when the final Effect instance attempted to load
 4. `getdspinfo 0` Returns detailed info about the DSP chip with the given index. By executing this command you can figure out whether particular chip is in use currently, which plug-ins are instantiated there, how many resources they consume and how many resources are still available.
- Figure 2: Info about the DSP chip with the given index.

12.45.3.3 DAE dish tips

- With the standard configuration, the system's [AAX](#) plug-ins folder will be used by DSH and DTT. To override this, create a folder named "Plug-Ins" next to the DTT and CommandLineTools directories. While that directory exists, AAE will only scan the plug-ins in the new Plug-Ins folder.

12.45.4 Basic plug-in tests

There are some basics tests that can be performed for AAX plug-ins in DSH. Among them are instantiation test, measuring of amount of processor cycles that DSP plug-in may consume on different settings, cancellation test and so on.

12.45.4.1 Cycle count performance test

Use the `DAE.cyclessshared` command in the DAE dish to profile a DSP algorithm's cycle count performance. This command measures both the shared and the per-instance cycles used by a plug-in, both of which must be reported to the host. This command also includes the option to load a custom plug-in preset so that various algorithm code paths may be exercised. It is important to report the maximum possible number of cycles that the plug-in may need, so that it had enough resources, even in the worst case. Otherwise one can obtain noise and clicks in the output audio on the extreme plug-in's settings.

The full syntax of this command is as follows: `cyclessshared <index -- spec -- {key:value, key:value, etc.}>` `index` - Index of the plug-in as listed by the `DAE.run` command `spec` - Plug-in ID triplet in array, e.g. ['AVID', 'DmGn', 'DGDt'] `key:value` hash options: `idx` - Index of the plug-in `spec` - Plug-in ID triplet array `run_cached`: `<true -- false>` - Whether to use cached code when measuring. Defaults to false. `load_preset`: `<filename>` - Load the specified preset for each instance before measuring performance `adjust_controls`: `<true - false>` - Randomly change the plug-in's parameter state before running the test

Examples:

- `cyclessshared 21`
- `cyclessshared ['AVID', 'DmGn', 'DGDt']`
- `cyclessshared {spec: ['AVID', 'DmGn', 'DGDt'], load_preset: "mySettings.tfx"}`
- `cyclessshared {spec: ['AVID', 'DmGn', 'DGDt'], adjust_controls: true}`
- `cyclessshared {idx: 21, run_cached: true}` *Do not use cached measurements for reported cycle counts!*

Normal output of this command should look like this:

Figure 3: Normal cyclessshared command output.

Sometimes during the development process it may happen that this test fails, and the reason of such a failure can be different:

1. Plug-in exceeded the DSP chip's memory limit
Figure 4: Plug-in exceeded the memory limit.

2. Plug-in exceeded the processors cycles budget

- The number of instance and shared cycles looks acceptable, but expected number of plug-in's instances that can be instantiated on the chip/card at different sample rates is zero. Resultant cycle count can be used for calculating how much plug-in has exceeded the limit, and how much it should be optimized.

Figure 5: Plug-in exceeded the processor cycles budget.

- If plug-in exceeds the processor's cycles budget too much, then cyclessshared test will most likely output the warning that is highlighted with orange color on the screenshot below. Also the number of both instance and shared cycles will be shown as zero or one.

Figure 6: Plug-in exceeded the processor cycles budget very much.

3. Plug-in processing is not balanced.

If some big parts of code, which do not really depend on the specific plug-in settings, are located under condition structures, and if they make plug-in to do more calculations in one case and less in another case, then that means that plug-in's processing is not balanced. This may cause some problems, because it is hard to predict when this or that condition may become true and how much the amount of processor's cycles that plug-in needs will increase. So it is better to remove such conditional blocks and to perform those calculations every time, even if their result is not really needed in particular cases.

To indicate such situations the correlation coefficient can be used. If its value close to zero, then plug-in has the described problems.

Figure 7: Plug-in processing is not balanced.

12.45.4.1.1 Performance profiling and test signals Some algorithms' performance characteristics are program-dependent, and in such cases use of the `cycles` command alone may not be sufficient. To route a test signal to your plug-in while measuring cycles, use of the `cycles` command along with the `load_wav_file` command in the DAE dish. The basic approach is as follows:

- Use single-buffer manual processing, rather than continuous
- Split your test signal into several pieces, with each piece to be processed using different settings
- Loop on:
 - Load or adjust the PI's settings,
 - process the next piece of audio while measuring cycles

Example:

1. `piproctrigger manual` set to single-buffer processing
2. `load_wav_file "testaudio_pt1.wav"`
3. `load_wav_file "testaudio_pt2.wav"`
4. `load_wav_file "testaudio_pt3.wav"` load audio buffers; take note of return ...
5. `run <mypluginIndex>`
6. `load_settings "mySavedSettings.tfx"` load the settings OR `control [1,24]` # set controls directly
7. `cycles b1` measure cycles while processing first file
8. `load_settings "mySavedSettings2.tfx"` load the next settings
9. `cycles b2` measure cycles while processing second file ... etc.

12.45.4.2 Cancellation test

When porting plug-ins from RTAS to AAX platform, or from 32-bit to 64-bit architecture, or from Native to DSP, it may be useful to compare output of two plug-in's versions to make sure that it is still the same and nothing has been broken. For this purpose DSH cancellation test can be used.

In the simplest case, when both plug-ins are present in the same version of Pro Tools (Native and DSP version of the same plug-in for example), then `diff` command can be used to perform the test: `diff [<spec1>, <spec2>, <frames>]` which reports the peak difference in the output amplitude of plug-ins `<spec1>` and `<spec2>` after processing `<frames>` frames of a 1 kHz full-scale sine wave. The maximum difference will be provided in dB.

Another way to perform the cancellation test is to process audio with each plug-in separately manually and to compare the result after that. This scenario allows you to load custom input audio file and special plug-in settings:

1. `piproctrigger manual` This command should be run for DSP plug-ins before loading them. When this option is set, DSP plug-in will start process audio only after `piproc` command is called. Otherwise it will start processing right after the instantiation process.
2. `run 81` Loading plug-in
3. `load_settings "/Users/settings/pitch_settings.tfx"` Loading settings file
4. `load_wav_file "/Users/audio_files/mono_file.wav"` Wav file will be loaded in a buffer, or in several buffer if it has more than one channel. Command will output references to the buffers, like `b1`, `b2` ...

Note

It is not recommended to choose very long audio files for the DSP processing, since the test is very slow, and processing of 10 sec audio file may take up to 1 min depending on the complexity of the plug-in's algorithm.

5. `bclone b1` The easiest way to create the output buffer of the same size is to copy the input buffer. Command will also output references to the resultant buffers.
6. `piproc [b1, b2]` Command which actually is doing passes the input audio through the current plug-in, and is recording the result to the output buffer (which is b2 here). For stereo plug-in command will look like `piproc [[b1,b2],[b3,b4]]`
7. `bfsave [b2, "/Users/saved_buffer"]` Output buffer can be stored to the disk. This may be needed for the cases, when one wants to compare the output of plug-ins, which can not be loaded in the same instance of the DSH. So the output of one plug-in can be saved to disk and then loaded later in the another instance of DSH. Also output buffer can be saved as .wav file with `save_wav_file` command.
8. `bflload "/Users/saved_buffer"` Saved buffer can be loaded again by this command. It will output the reference to the newly created buffer.
9. `bacmp [b1,b2]` This command will compare the contents of the buffers b1 and b2. So it can compare the output buffers of two plug-ins and thus make the cancellation test.

12.45.5 Debugging and tracing in DSH

DSH provides a lighter-weight debugging environment than the full Pro Tools application. So it should be easier to step through the description code of the plug-in there, rather than in PT, because DSH does significantly less initialization work than PT during the loading process.

Also DSH is very useful in situations, when one wants to debug the plug-in's algorithm on a specific audio buffer. The only way to follow plug-in's algorithm work step-by-step on the certain piece of audio is to debug the `piproc` command.

DSH supports tracing, which is based on Avid's DigiTrace. To enable trace logs in DSH, one should create a `dsh.digitrace` config file for it and put it next to `dsh` executable file. It can be the same as `.digitrace` file for the Pro Tools. DSH has built-in commands to generate a DigiTrace config file. The `clear_trace_config` command creates (or clears if it already exists) a DigiTrace config file. The `enable_trace_facility` command enables logging of a specified facility/priority pair.

Note

On the Mac, DigiShell must be relaunched before a new DigiTrace configuration will take effect.

12.45.6 Scripting interface and batch profiling

DigiShell can be scripted using DishTestTool, a Ruby-based command line tool. More details can be found in [DTT Guide](#).

Collaboration diagram for DSH Guide:



12.46 DTT Guide

How to automate different test scenarios for DSH.

12.46.1 Contents

- [What is DTT](#)
- [How to run tests and suites in DTT](#)
- [Writing DTT scripts](#)
- [Logging in DTT and debugging DTT scripts](#)
- [Working with DTT test suites](#)

12.46.2 What is DTT

DishTestTool (DTT) is a Ruby-based command line tool, which provides the ability to script and thus automate DSH test scenarios. It is included in the DigiShell Tools package in the /DTT directory.

Note

Ruby is installed by default on macOS. On Windows, you will need to install Ruby and add it to your PATH variable manually. For information regarding Ruby version compatibility with a specific build of DTT, see the ReadMe.txt file in /DTT/sources.

The DTT folder consists of:

- *Sources*
 - DTT core
 - *scripts* folder - place all DTT script files here
By default, this folder includes a few example scripts demonstrating basic DTT operations and plug-in testing steps:
 - * *DSH_SigCancellation.rb* - script for the cancellation test
 - * *DSH_TI_CycleCounts* - script for performing cycle count test
 - * *SuiteGenerator.rb* - generates suites for the DSH_SigCancellation and DSH_TI_CycleCounts tests
 - *suites* folder - place your DTT suite files here
- *run_test.command* (on Mac) or *run_test.bat* (on Windows) - command file for running tests
- *run_irb.command* (on Mac) or *run_irb.bat* (on Windows) - command-line interpreter

12.46.3 How to run tests and suites in DTT

`run_test` is the main DTT execution program. `run_test` is able to execute Ruby scripts which have been placed in the `scripts` folder within the DTT directory.

- `run_test.command -l` - lists all the available scripts and suites
- `run_test.command 1` - runs test by number
- `run_test.command DSH_SigCancellation` - runs a test by name. Pay attentions that the name of test should be without (!) extension.
- `run_test.command -script DSH_SigCancellation -a sample_rate=48000 -a threshold=-80` - runs a test with test script arguments, which are specified using the `-a` option

Figure 1: running DTT tests.

For more information about script arguments, see [Describing and using input arguments of the script](#)

12.46.4 Writing DTT scripts

Most of the DTT scripts require `DigiShell`, which allows them to run `dsh` and execute different `dsh` commands. Each script should be represented in the form of class, which inherits `Script` class, and also each script must have at least two elements: `self.inputs` section, where all the input arguments of the test should be described, and `run` method, which is the main body of your script.

```
require 'DigiShell'

class ScriptSample < Script
  def self.inputs
    return {}
  end

  def run
    return pass("Well, it didn't explode. So that's something.");
  end
end
```

Listing 1: Skeleton of the script

12.46.4.1 Describing and using input arguments of the script

The available parameters and their values for a script are listed in the static `self.inputs` routine. Input arguments must be organized in the form of a hash map which is returned from this routine.

```
def self.inputs
  return {
    :sample_rate => [44100, [44100, 48000, 88200]],
    :path_to_tfx => ['none'],
    :threshold => [-96],
  }
end

@dsh.init_dae(sample_rate)
```

Listing 2: Describing input arguments for the script and using them

Hash entries should be in the following format: `:arg_name => [default_value, [range of allowed values]]`

These arguments can be used then by just calling them by name, like in the example above with `sample_rate` argument.

12.46.4.2 Writing body of the script

The body of the script must be enclosed in the body of the `run` method of the script class. As far as most DTT tests need DSH, in the example below it shows how to create a DSH instance in the script and how to use it then. DSH instance can be created with `DigiDShell.new` method, which requires `DigiShell` module, as has already been said. Then all the DSH commands become available as methods of the DSH instance, and input arguments can be passed to these commands as input arguments for the methods, i.e. in parentheses `dsh.load_dish("DAE")`. Also it's recommended to handle possible exceptions that may occur during the execution of the code, and to make sure that DSH has been closed, if it was instantiated on the moment of the failure.

```
def run
  begin
    dsh = DigiShell.new(target)
    dsh.load_dish("DAE")
    dsh.init_dae(sample_rate)
    dsh.close

    return pass("Well, it didn't explode. So that's something.")

  rescue Exception => e
    # make sure to close down dsh before returning
    if (dsh)
      dsh.close
    end

    return fail(e)
  end
end
```

Listing 3: Example of the body of the script.

12.46.5 Logging in DTT and debugging DTT scripts

DTT tool has logging, and all the logs collected in the Logs folder, which is located in root directory of the DTT. DTT creates a separate folder for each test and names these folders with the corresponding names + the time when the particular test has been executed. For example:

DSH_SigCancellation_20131225_185146_0001

Inside each folder there are several log files:

- `xxx.html` – contains info about input & output of the test in a fancy form (tables)
- `xxx_c.txt` – contains the list of the DSH commands that have been executed
- `xxx_d.txt` – DSH output
- `xxx_i.txt` – info about your system
- `xxx_l.txt` – standard output
- `xxx_v.txt` – verbose output

12.46.5.1 Interactive mode

There is an option to run DTT in interactive mode using interactive ruby shell (`irb`). When running in this mode, DTT creates a shell which is an extended version of the standard Ruby interpreter. Besides the standard functionality, it knows how to work with DTT classes and can give hints on their methods. In particular, the DTT interactive mode shell knows how to work with `DigiShell`.

To run DTT in interactive mode, go to the DTT folder and launch the `run_irb` program. At this point you will send ruby commands to dsh through the pipe in YAML format:

```
t = Target.new # creates an instance of Target. In this case target is a local machine, though we have a
               # possibility to run test on remote machine.
dsh = DigiShell.new(t) # creates an instance of DigiShell(aka launching dsh binary)
dsh.load_dish('DAE') # Loads 'DAE' dish
dsh.help('init_dae') # Requests help from dsh for 'init_dae' command
dsh.init_dae
plugins = dsh.run #returns an array of plug-ins and writes them to plugins var.
plug-ins[0] #reaching first plug-in from the list
#... whatever you want to do
```

Listing 4: Running DTT in interactive mode

12.46.6 Working with DTT test suites

Suites are files which contain the list of DTT scripts that should be run, and parameters for these tests. These files should be created in YAML format. The list of the tests should be preceded by `tests:` line. Then tests to be run should be described as a map with the following members:

- `name:` - name of the test
- `enabled:` - determines whether test will be run or skipped
- `args:` - input parameters of the test

The input parameters of the test should be organized as a hash map. That means that all keys should start with ":" and look like `:plugin_spec:`.

Also suite may contain a section with the general parameters like:

- `verbose:` `false` which determines whether the output of the test in the console will be verbose or not.
- `timeoutFactor:` `"16.0"` which defines the time period, after which test will exit in case it stuck on the execution of the certain piece of code.

Example of the suite:

```
suitesettings:
  verbose: false
tests:
#
# Cycle counting test
#
- name: DSH_TI_CycleCounts
  enabled: true
  args:
    :plugin_spec: 'Digi,Pich,Psmm'
    :sample_rate: 48000
```

Listing 5: Example of the DTT test suite

Note

All the suites files should have an extension `.gss`

12.46.6.1 Autogeneration of the suites

Sometimes it is necessary to generate the suites for the particular script for all the plug-ins from the bundle and/or for different sample rates. In this case instead of the copy-paste, which may lead to some mistakes and erratums, `suitesgenerator` can be used. This is a special script, which takes as arguments the name of the script(s), for which the suites should be generated, and the list of their input parameters. Strange as it may sound, this data should be formed as a suite. Script itself is available as `SuiteGenerator.rb` along with other scripts in the DTT.

Note

SuiteGenerator.rb can generate suites only for the two tests: DSH_SigCancellation test and DSH_TI_CycleCounts test

Here is the example of how to use this script to generate the suites for all the plug-ins from the 'D-Verb' bundle for all the sample rates the cancellation test AAX Native vs AAX DSP, and for the cycle counts test:

```
tests:
# Generate suite for Cancellation test: AAX Native vs AAX DSP
- name: SuiteGenerator
  args:
    :plugin_name: 'D-Verb'
    :path_to_audio_files: /Volumes/G_Audio/GS_Test_Resources/audio/
    :path_to_presets: /Volumes/G_Audio/GS_Test_Resources/PL_Settings/
    :test_script: DSH_SigCancellation
  enabled: true
# Generate suite for Instance Count test
- name: SuiteGenerator
  args:
    :plugin_name: 'D-Verb'
    :path_to_presets: /Volumes/G_Audio/GS_Test_Resources/PL_Settings/
    :test_script: DSH_TI_CycleCounts_se
  enabled: true
suitesettings:
  verbose: false
  timeoutFactor: "16.0"
```

Listing 6: Example of how to use SuiteGenerator script for generating suites for all the plug-ins from the 'D-Verb' bundle for the all sample rates for the cancellation and for the cycle counts test.

To generate the suites, one should run this suite for the SuiteGenerator as an ordinary suite by executing the `run_test.command <the name of the suite for the SuiteGenerator>` command. All the suites will be generated into the single file, which will be located inside the suites folder, and will be called like:

`dspVSnative(optional)_<the name of the plug-in>_<the name of the test>.gss`

Examples:

- TRIM_DSH_TI_CycleCounts.gss
- dspVSnative_TRIM_DSH_SigCancellation.gss

Collaboration diagram for DTT Guide:



12.47 Extensions

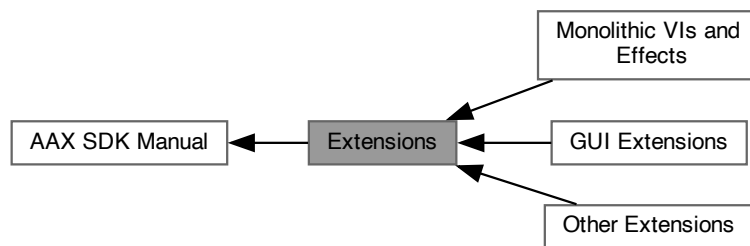
12.47.1

Extensions to the AAX SDK.

Documents

- [GUI Extensions](#)
GUI Extensions for the AAX SDK.
- [Monolithic VIs and Effects](#)
- [Other Extensions](#)

Collaboration diagram for Extensions:



12.48 GUI Extensions

GUI Extensions for the AAX SDK.

12.48.1 About the SDK's GUI Extensions

The code and projects in the SDK's Extensions/GUI/ directory demonstrate how to extend the AAX SDK's GUI programming interface using a variety of popular GUI frameworks, including:

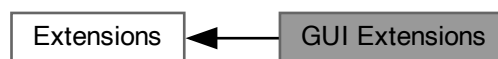
- Native Cocoa
- Native Win32
- VSTGUI
- JUCE

These projects do not represent core functionality of the AAX SDK, but rather they serve as examples of how plug-in GUIs can be written to the AAX specification using a variety of different approaches.

12.48.2 Notes

- The VST and JUCE GUI Extension library projects use a macro value to resolve file paths to the installed framework directory. This macro is defined in a Visual Studio property sheet on Windows and as a custom project variable on Mac. Because this macro will not be resolved on Mac until compilation, the Xcode GUI will not be able to find the included files. However, the projects should build successfully once the macros are updated to point to the correct directory.
- The JUCE GUI Extension code in this SDK was written using version 1.51 of the JUCE framework
- Several VSTGUI-based headers with an "_ext" filename are provided along with the SDK's GUI Extensions code. These headers are slightly modified versions of the corresponding headers that are distributed with VSTGUI. The headers have been modified for use with our example plug-ins because we had several problems when using the VSTGUI SDK for 64 bit. For example, we encountered conflicting typedefs/redefinitions like `int32_t` etc., which are preprocessed out of the `vstguibase_ext.h` file. These headers should not be required to build a 32-bit plug-in and they were only added in our transitional AAX SDK, version 1.5.

Collaboration diagram for GUI Extensions:



12.49 Monolithic VIs and Effects

Extension of the [AAX_CEffectParameters](#) class for monolithic VIs and effects.

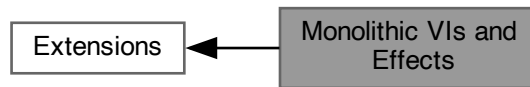
This extension to [AAX_CEffectParameters](#) adds some conveniences for Virtual Instrument (VI) plug-ins and for other plug-ins that use a monolithic processing object, i.e. an object that combines state data with the audio render routine in a single object.

- The [RenderAudio](#) method provides a direct audio processing callback within the data model object. Perform all audio processing in this method.
- The [StaticDescribe](#) method establishes a generic MIDI processing context for the Effect. Call this method from the plug-in's [Description callback](#) implementation.
- The [AddSynchronizedParameter](#) method provides a mechanism for synchronizing parameter updates with the real-time thread, allowing deterministic, accurate automation playback. For more information about this feature, see [Fixing timing issues due to shared data](#)

Note

This convenience class assumes a monolithic processing environment (i.e. [AAX_eConstraintLocationMask_DataModel](#).) This precludes the use of [AAX_CMonolithicParameters](#) -derived Effects in distributed-processing formats such as AAX DSP.

[AAX_CMonolithicParameters](#) Collaboration diagram for Monolithic VIs and Effects:



12.50 Other Extensions

12.50.1

MIDI logging utilities

- void [AAX::AsStringMIDIStream_Debug](#) (const [AAX_CMidiStream](#) &inStream, char *outBuffer, int32_t in↔ BufferSize)

Filesystem utilities

- bool [AAX::GetPathToPlugInBundle](#) (const char *iBundleName, int iMaxLength, char *oModuleName)
Retrieve the file path of the .aaxplugin bundle.

12.50.2 Function Documentation

12.50.2.1 AsStringMIDIStream_Debug()

```
void AAX::AsStringMIDIStream_Debug (
    const AAX\_CMidiStream & inStream,
    char * outBuffer,
    int32_t inBufferSize )
```

Print a MIDI stream as a C-string

Sets an empty string in release builds

12.50.2.2 GetPathToPlugInBundle()

```
bool AAX::GetPathToPlugInBundle (
    const char * iBundleName,
    int iMaxLength,
    char * oModuleName )
```

Retrieve the file path of the .aaxplugin bundle.

Parameters

in	<i>iBundleName</i>	<ul style="list-style-type: none"> • macOS: The <code>CFBundleIdentifier</code> value set in the plug-in's .plist file • Other: This parameter is ignored
in	<i>iMaxLength</i>	
out	<i>oModuleName</i>	A preallocated buffer of size <code>iMaxLength</code>

Collaboration diagram for Other Extensions:



12.51 Supplemental Information

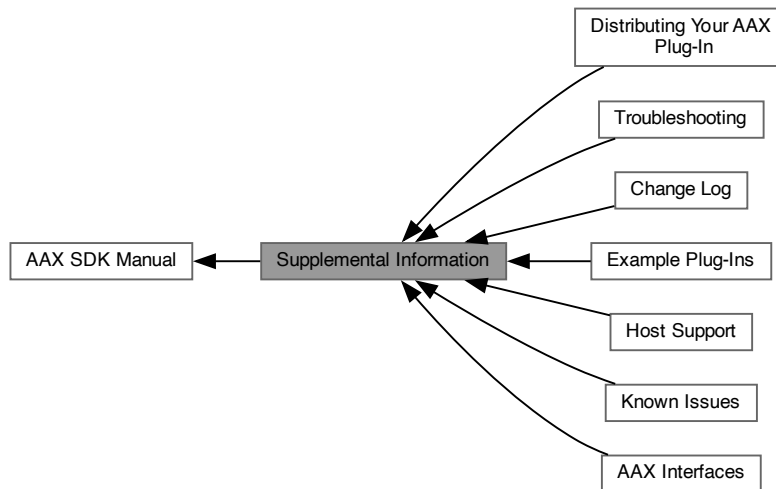
12.51.1

Supplemental documents beyond the scope of the AAX SDK.

Documents

- [Troubleshooting](#)
How to solve common issues.
- [Distributing Your AAX Plug-In](#)
Details about packaging and distributing your AAX plug-ins.
- [AAX Interfaces](#)
Full list of AAX interfaces.
- [Host Support](#)
Supported features in each AAX host.
- [Known Issues](#)
A list of known bugs affecting AAX plug-ins.
- [Change Log](#)
Changes between AAX SDK versions.
- [Example Plug-Ins](#)
Descriptions of the SDK's example plug-ins.

Collaboration diagram for Supplemental Information:



12.52 Troubleshooting

How to solve common issues.

12.52.1 Contents

- [Plug-In Fails to Load in Shipping Pro Tools](#)
- [Plug-In Causes Audio Streaming Errors](#)

12.52.2 Plug-In Fails to Load in Shipping Pro Tools

If your plug-in fails to load in shipping Pro Tools with the message "The following plug-ins failed to load because they are not valid 64 bit AAX plug-ins" then the most likely reason is that the plug-in does not have a valid digital signature.

Your AAX plug-ins will not be compatible with shipping versions of Pro Tools until they are digitally signed using tools provided by PACE Anti-Piracy, Inc. As an AAX developer you can receive these tools free of charge. Read the [Digital signature](#) section of the [Pro Tools Guide](#) to learn about the digital signing requirements for compatibility with Pro Tools.

To verify whether this failure is due to an invalid digital signature vs. some other library loading failure, check the Pro Tools [log file](#). A failure caused by an invalid digital signature will result in log lines like the following:

```

Sys_PACE::GetDigitalSignature - looking for Eden dsig for path "/Applications/ProTools/Plug-Ins/DemoGain_example.aaxplugin/ 0
Sys_PACE::GetDigitalSignature - dsig error name /Applications/ProTools/Plug-Ins/DemoGain_example.aaxplugin/ 0
legacy Dsig check disabled??
Sys_PACE::GetDigitalSignature - did NOT get valid dsig /Applications/ProTools/Plug-Ins/DemoGain_example.aaxplugin/ 0
Plug-In Binary "DemoGain_example.aaxplugin" failed to load with err = -7054.
Plug-In Binary "DemoGain_example.aaxplugin" 1.0 : Failed to load.
  
```


Another way to check whether a plug-in's digital signature is invalid is to test the plug-in in a Pro Tools developer build or with the [DigiShell](#) utility. If the plug-in successfully loads and runs in these tools but not in a shipping build of Pro Tools then it is very likely that the problem is in the plug-in's digital signature.

If you are having an issue running the signing tools then please check this list of the most common failure points:

1. Bad command line arguments for `wraptool`
2. An invalid developer certificate
3. An expired developer certificate
4. The Eden Tools license is not activated to your iLok USB key
5. Your code signing certificate is not installed on your iLok USB key
6. For Mac, the Xcode command line tools are not installed on your signing system
7. The plug-in bundle itself is malformed and will not load
8. The plug-in bundle is being modified at some point after being signed, thereby invalidating its digital signature

If a digital signature was successfully applied to an AAX plug-in at one point in the build process but now the plug-in fails to load due to a bad signature then the most likely reason is that someone or something has altered the signature or the contents of the `.aaxplugin` bundle thereby invalidating the signature. The most common reason for a digital signature to become invalidated is that something is changed within the `.aaxplugin` bundle when moving between different systems or when archiving/unarchiving.

Several things can cause this kind of signature invalidation. Here are some examples:

- If symlink are not preserved when copying the plug-in
- If there was some actual tampering of the plug-in after the build
- If there is corruption of the plug-in binary itself
- If the `.aaxplugin` bundle contains one or more file names with exotic characters which change representations when moved between filesystems with different character encoding schemes

Note that the AAX digital signature covers the entire `.aaxplugin` bundle so any actions which affect the contents of this bundle in any way after signing will invalidate the bundle's digital signature.

If the failure is occurring on an isolated system then replacing the `.aaxplugin` which has an invalidated signature with an original, untampered copy (e.g. via a reinstall) should resolve this issue.

If the failure is occurring only on certain systems then try archiving and copying the failing plug-ins back to a system where the plug-in loads successfully then comparing the archived copy to a known successful copy to see if there are any differences to the file names or binary contents of the files within the bundle.

12.52.3 Plug-In Causes Audio Streaming Errors

See also

[Real-time performance](#)

The algorithm callback in audio plug-ins is executed within a complex real-time environment, often with tight deadlines for not only the plug-in itself but for other plug-ins participating in the same processing chain. The real-time threading model is managed outside of the plug-in and may be different across different plug-in hosts and formats. Sometimes, things go wrong and a deadline is missed.

Figure 1: Processing thread utilization during a sporadic audio streaming error

This can happen naturally when the system is loaded to capacity and the CPU simply does not have time to complete all of the work required by the plug-in algorithm routine before the processing deadline. Most often, however, audio processing errors occur in situations when there ought to be more than enough time to do all of the work required by the plug-in. As shown in the image above, the real-time threads can appear to have low CPU utilization and plenty of overhead, then suddenly a deadline is missed. What happened?

In most cases, audio engine errors occur when a single plug-in instance significantly overruns the processing deadline. The instance usually processes quickly in prior executions and does not give any indication of impending doom.

Figure 2: Call execution times for three plug-in instances in a chain

There are many reasons why this can happen. One excellent tool for evaluating these kinds of failures on macOS is the `ktrace` utility. This utility collects system calls, thread interactions, and backtraces similar to Instruments data in a simple command line tool. This can provide a detailed view of the state of the system leading up to an audio streaming error, and can be used to capture logs on any Mac system, even those without special developer tools installed. Once the tool is running there is a minimal performance impact. Avid provides a `ktrace` capture utility for use with Pro Tools that can trigger captures based on Pro Tools audio engine errors. You can download this tool from the AAX SDK downloads area.

Figure 3: Beginning in Pro Tools 2021.6, this dialog is presented when a plug-in significantly overruns its deadline

Here are some of the culprits that Avid has found when investigating these kinds of performance issues using `ktraces` and similar utilities. Use these examples as a guide for the kinds of things to watch out for in your plug-ins, especially when you hear reports that your plug-ins may be triggering sporadic audio engine errors.

- System calls, C++ library calls, and other language features' call synchronization

When tracking down intermittent performance issues, watch for any calls into the standard C++ library or any use of higher-level language features that are not explicitly designed for real-time use.

You should never trust that STL and C++ library implementations are safe to use in a real time context. Of course STL containers are not thread safe, but in real-time code you should avoid all use of C++ library functions, not just containers, unless you are certain that the library implementation you are using will have no performance related side effects.

For example, in some macOS versions `std::clock` will take a kernel mutex, causing an unexpected priority inversion with any lower-priority thread using `std::clock` at the same time.

Furthermore, the runtime features of higher-level languages are often not designed for running in real time. Objective-C and Swift messaging calls can take locks and should never be used in a real time thread and other languages' features are similarly out of your control as a developer. Avoid them in your algorithm logic.

- Other components and libraries

Similarly, any third-party component or library should not be used from the real-time thread unless it is explicitly designed for use in this context. It can sometimes be difficult to track calls into library code, especially if the library use is not isolated to a particular function in the plug-in such as its graphics or signal processing. Be particularly careful to isolate the plug-in's algorithm logic when using general-purpose libraries that serve multiple functions in a plug-in or when using objects that combine multiple functions.

- Synchronization within the plug-in

Any synchronization of data access within a plug-in must be performed in a way such that the real-time algorithm can never block on a resource held by a lower priority thread.

In an AAX plug-in, parameter changes are applied on a different thread than audio processing. AAX includes a system for synchronizing delivery of parameter updates to the algorithm at the correct time without requiring any synchronization by the plug-in, and you are strongly encouraged to understand the details of how [parameter updates](#) work, in particular how the [parameter update timing](#) is achieved and how to implement proper timing and synchronization even in a plug-in that does not use a standard decoupled algorithm callback.

Be particularly careful if you do not use this system for decoupling and if instead your plug-in shares access to the same data between its algorithm and other logic, even if you are using a third party framework. If your plug-in triggers sporadic deadline misses in the host engine then this can be a productive area to inspect

- File access

File access can accidentally creep into the logic executed by a plug-in algorithm, causing random but severe timing failures. Check to make sure that your plug-in cannot possibly trigger any logging to a file from the audio processing thread. Even having a file handle open on the processing thread can cause performance problems: the OS may trigger a flush on such an open file during a filesystem synchronization event, causing the thread to block despite there being no explicit file I/O operations performed.

- Virtual memory faults

If your plug-in frequently accesses large amounts of data then be sure to check for possible virtual memory faults in any logs concerning audio buffer overruns. Avoid any access to paged memory or files from within the plug-in's real-time code.

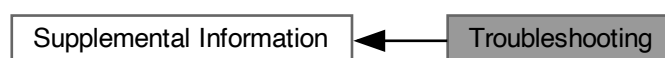
- Memory allocation

Memory allocation is the classic example of what not to do in a real-time thread, yet it can be quite difficult to track down. There are no guarantees for the time that a `malloc` call may execute; the call may even need to page memory in from disk in order to complete. It can be very difficult to determine whether any particular call can result in memory allocation, and this underscores the total control that you must take over your plug-in's algorithm logic. Every operation performed within the audio processing thread must be guaranteed to be free of memory allocation, which requires a deep understanding and control over the implementation of all methods called from the algorithm.

- User authorization and copy protection

Some copy protection schemes will helpfully scatter authorization checks throughout your code, relieving you of that chore. Be sure to exclude your plug-in's real-time logic from this process. Authorization checks are complex and can take multiple milliseconds to complete, or more if they involve external licensing hardware or contact with an external server.

Collaboration diagram for Troubleshooting:



12.53 Distributing Your AAX Plug-In

Details about packaging and distributing your AAX plug-ins.

12.53.1 Contents

- [The finishing touches](#)
- [Building your plug-in installer](#)
- [Testing your plug-in](#)
- [Selling your plug-in](#)

12.53.2 The finishing touches

You've completed your main development work and your new AAX plug-in is nearly ready to ship! Now it's time to put the polish on your release.

12.53.2.1 Check and finalize page tables

After development has completed on your plug-in, we recommend that you check and finalize the plug-in's page tables using the [Page Table Editor](#) tool. It can be easy to forget to update the plug-in's page tables after making changes to the plug-in's list of parameters or to other aspects of the plug-in during development. To check for problems, open and view the plug-in's page tables for every layout in the editor app. Verify that the plug-in parameters are arranged properly for each control surface and that the list of available parameters in each layout is correct.

Correct and complete page tables are an important part of the user experience for many AAX plug-in users, and your users will appreciate your attention to detail here!

12.53.2.2 Create factory presets

Each AAX plug-in may be bundled with a set of factory presets. These presets will be made available to users through the host application's plug-in preset management UI.

Plug-in factory presets are stored as .tfx settings files. These files can be generated from any AAX host application which supports plug-in preset management. For example, in Pro Tools it is possible to create a new .tfx settings file by following these steps:

1. Create an instance of your plug-in in a Pro Tools session
2. Manually apply the desired preset settings
3. Choose "Save Settings As..." from the Presets drop-down menu in the plug-in window header

Once you have saved your desired factory presets as .afx files onto your system you can package them with your plug-in bundle in *.aaxplugin/Contents/Factory Presets. Any presets found in this directory will be copied to the plug-in settings location for the running instance of Pro Tools when Pro Tools scans the plug-in on launch. See [.aaxplugin Directory Structure](#) for more information about supported sub-directories within the .aaxplugin bundle.

The feature for automatically copying factory presets from the .aaxplugin bundle to the plug-in settings directory on the user's system is supported by Pro Tools 11 and later and by all versions of Media Composer with AAX plug-in support.

Plug-in installers for 32-bit plug-ins supporting Pro Tools 10.3.5 and earlier must copy the settings to the plug-in settings folder when the plug-in is installed.

These are the paths for plug-in settings used by Pro Tools and Media Composer versions which support 32-bit AAX plug-ins:

- Mac: /Library/Application Support/Digidesign/Plug-In Settings
- Win: C:\Program Files(x86)\Common Files\Digidesign\DAE\Plug-In Settings

The default paths for plug-in settings used by Pro Tools and Media Composer versions which support 64-bit AAX plug-ins are provided below. However, you should **not** use these paths in your installers since they may be customized using the host application's preferences (for example, the "User Media and Settings Location" preference in Pro Tools.) Instead, use the Factory Preset bundling system described above for installing presets for 64-bit plug-ins.

Default plug-in settings locations for 64-bit [AAX](#) plug-in hosts:

- Mac: ~/Documents/Pro Tools/Plug-In Settings
- Win: C:\[user folder path]\Documents\Pro Tools\Plug-In Settings

For more information about using plug-in presets in the various AAX hosts, see the following pages in the documentation for each host:

- [Pro Tools](#)
- [Media Composer](#)
- [VENUE](#)

12.53.2.3 Sign your plug-in

Pro Tools requires that all AAX plug-ins be signed with a digital signature. The certificate authority for this signature is PACE Anti-Piracy, Inc. and all AAX plug-ins for Pro Tools must be signed with the digital signing tools from PACE. See the [Digital signature](#) section in the [Pro Tools Guide](#) for more information about this requirement.

12.53.3 Building your plug-in installer

Your plug-in installer should place all .aaxplugin bundles into the system's AAX Plug-Ins directory:

- macOS: /Library/Application Support/Avid/Audio/Plug-Ins
- Windows (32-bit plug-ins): C:\Program Files (x86)\Common Files\Avid\Audio\Plug-Ins
- Windows (64-bit plug-ins): C:\Program Files\Common Files\Avid\Audio\Plug-Ins

This directory is searched recursively, so AAX plug-ins may be installed into sub-directories. For example, you may install all AAX plug-ins into a new sub-directory labelled with your manufacturer name.

12.53.3.1 Installing Track Presets

The Track Presets feature in Pro Tools allows users to recall entire tracks, or entire sets of tracks, and to add specific track data such as insert chains, sends, and routing. For example, if a user doesn't know in advance what vocal chain they may want to use, they can begin tracking, and then instantiate a whole set of inserts with stored settings from an existing track preset by clicking on an insert selector and finding that preset.

You are encouraged to create your own track presets and provide them to users in your installers. For example, if you sell plug-in bundles then you may wish to provide users with Track Presets demonstrating useful combinations of multiple plug-ins from the bundle, or if your plug-ins involve some "boilerplate" routing configuration then you can provide a multi-track Track Preset with this routing already established.

Installation Location

Track Presets are stored in the Pro Tools documents folder. Use these locations for default installation

- Mac ~/Documents/Pro Tools/Track Presets
- PC: C:\Users\[username]\Documents\Pro Tools\Track Presets

This location is indexed automatically by Pro Tools.

All of the Track Preset files which you install should be added to a folder with the name of your company. This will ensure that your Track Presets appear as expected in the preset menus in Pro Tools:

- *Pro Tools Documents Folder*
 - /Track Presets
 - * · *Name of your company*

Tagging

A default tags dictionary is available from the [My Toolkits and Downloads](#) page at avid.com. These are not the only tags you can use, but any of these that you do use will be increasing the value and usability of the default set included with Avid products. Using this shared dictionary will ensure that your users can quickly find your Track Presets. Workflow Considerations

- Audibility
 - If you want a track to be heard automatically then route that track to the Monitor Path. If a user is using a Monitor Path the track preset will be instantiated and audible immediately.
- Track Data to Recall
 - In most cases a Track Preset will be created exactly as a user wants to recall it. The available Track Data to Recall from a preset is quite broad though, so you should consider what default import settings make the most sense for each of your presets.
Here are some ideal default settings for a generic single track plug-in focused preset:
- Plug-in Format Conversion
 - Format conversion for plug-ins is designed to work if formats are enumerated correctly and available. This would take place for instance when recalling inserts from a stereo track preset to a 5.1 track preset - most often this should just work if your plug-in is available in all/most formats.
- Including Avid Stock Plug-ins

- If you wish to include any stock Avid plug-ins in your presets for any reason, stick to these plug-ins that are automatically installed by Pro Tools to be as sure as possible that your end user will be able to fully recall the preset:

- * *AutoPan; BF-76; Channel Strip; Click II; Dither; Down Mixer; D-Verb; Dynamics III; Eleven Lite; EQ III; InTune; Invert/Duplicate; LoFi; Master Meter; Maxim; ModDelay III; Normalize-Gain; Pitch Shift Legacy; Pitch II; RectiFi; Reverse/DC Removal; SansAmp PSA-1; SciFi; Signal Generator; Time Shift; Time Adjuster; Trim; VariFi*

The following Virtual Instruments are installed separately but come for free with paid Pro Tools versions:

- * *Boom; DB-33; Mini Grand; Structure Free; Vacuum; Xpand!2*

12.53.4 Testing your plug-in

The AAX Plug-In Burnthrough Grid document describes a number of test cases and workflows for multiple AAX plug-in hosts. This document is available for download as part of the AAX SDK Toolkit on the [My Toolkits and Downloads](#) page at [avid.com](#).

12.53.5 Selling your plug-in

12.53.5.1 Avid Marketplace

Avid may offer to sell your compatible products through our online store. We offer test tools and support services that will help you get your products to market with the highest quality whether you decide to offer them through our online store or independently. Registered developers can further register as Sellers, then work with Avid to add their solutions to the online store. Please visit your My Avid account and go to "My Developer Account" then to "Access Seller Portal" to explore this program or write to partners@avid.com for more information.

Get your AAX Plug-In ready for sale on Avid Marketplace by following these steps:

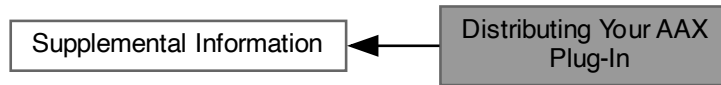
- *Explore the Avid Webstore* - Review the [Avid Webstore description](#) and learn about this valuable and expanding offering. E-mail us at partners@avid.com with your questions.
- *Sign up* - Register as a Seller (sometimes referred to as a "vendor") by following the link from the "My Developer Account" page and selecting "Access the Seller Portal."
- *Prepare your submission* - Gather the plug-in and other information required to onboard as described in the Onboarding FAQ. Your experience will be easier if you collect these items in advance.
- *Send us your Product* - Submit your products and other required information for testing and publication on the Avid Store!

12.53.5.2 In-App Purchase

In-App Purchase provides a direct path to purchase your products from directly within the AAX host application. For example, when a user opens a session which contains unavailable plug-ins, In-App Purchase can be used to prompt the user to purchase the plug-ins immediately.

See [this article](#) for more information about how to add support for In-App Purchase to your on-boarded AAX plug-ins. Additional documentation regarding In-App Purchase is available under the "In-App Purchase Tools" section of the [AAX SDK Toolkit](#) downloads page in your [avid.com](#) account.

Collaboration diagram for Distributing Your AAX Plug-In:



12.54 AAX Interfaces

Full list of AAX interfaces.

12.54.0.1 Interfaces Implemented by the AAX Host

These interfaces are implemented by the AAX Host. References to the host-managed objects are provided to the plug-in through accessor methods, most commonly [IACFUnknown::QueryInterface\(\)](#).

Class [AAX_IAutomationDelegate](#)

Class [AAX_ICollection](#)

Class [AAX_IComponentDescriptor](#)

Class [AAX_IController](#)

Class [AAX_IDma](#)

Class [AAX_IEffectDescriptor](#)

Class [AAX_IHostProcessorDelegate](#)

Class [AAX_IHostServices](#)

Class [AAX_IMIDINode](#)

Class [AAX_IPrivateDataAccess](#)

Class [AAX_IPropertyMap](#)

Class [AAX_ITask](#)

Class [AAX_ITransport](#)

Class [AAX_IViewContainer](#)

12.54.0.2 Interfaces Implemented by the AAX Plug-In

These interfaces must be implemented by the AAX plug-in. Default implementations are provided in the AAX Library via the `AAX_C` classes. Plug-in classes may inherit from the `AAX_C` classes to override the default behavior.

Class [AAX_IACFTaskAgent](#)

Class [AAX_IEffectDirectData](#)

Class [AAX_IEffectGUI](#)

Class [AAX_IEffectParameters](#)

Class [AAX_IHostProcessor](#)

12.54.0.3 Interfaces internal to the AAX SDK

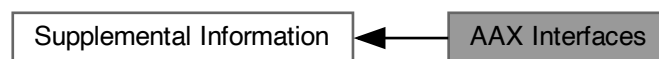
These classes and interfaces are used internally within the AAX Library. References to objects implementing these classes are never passed between the plug-in and the AAX Host, and the AAX Host has no knowledge of these classes.

Class [AAX_IParameter](#)

Class [AAX_IParameterValue](#)

Class [AAX_ITaperDelegateBase](#)

Collaboration diagram for AAX Interfaces:



12.55 Host Support

Supported features in each AAX host.

12.55.1 Host Support

These tables list AAX host support for various AAX interfaces, as well as support for general features. The tables include the version number for the earliest version of each Avid host software which supports the given interface or feature.

The earliest version of each host to support AAX plug-ins is:

- [Pro Tools](#) 10.0
- [Media Composer](#) 8.1
- [VENUE](#) 4.1

For more information about versioning in AAX, including how to check for host support of a particular interface, see [The Avid Component Framework \(ACF\)](#).

12.55.1.1 Platform Support

	Pro Tools	Media Composer	VENUE
AAX Native	10.0	8.1	<i>none</i>
AAX DSP	10.0	<i>none</i>	4.1
AAX Hybrid	11.0*	<i>none</i>	<i>none</i>
Offline processing (AudioSuite)	10.0	8.1**	<i>none</i>
ProcessProc / data model co-location	10.0	8.1	<i>none</i>
Monolithic topology	10.0	8.1	<i>none</i>
Native processor architecture	10: x86/i386 11+: x86_64	8.1+: x86_64	4.1: x86/i386 4.5+: x86_64
Compatibility with arm64/x86_64 FAT binaries on macOS	2021.10		n/a

Note

Pro Tools 11.0 supports AAX Hybrid processing for real-time plug-ins only. Support for AudioSuite processing for AAX Hybrid is supported starting in Pro Tools 11.1.

Media Composer 8.5 and higher support both multichannel and mono AudioSuite processing. Earlier versions of Media Composer support mono only.

12.55.1.2 Describe Interfaces

	Pro Tools	Media Composer	VENUE
AAX_IACFCollection	10.0	8.1	4.1
AAX_IACFComponentDescriptor	10.0	8.1	4.1
V2	11.0	8.1	4.5?
V3	12.8	<i>none</i>	5.6
AAX_IACFEffectDescriptor	10.0	8.1	4.1
V2	11.0	8.1	4.5?
AAX_IACFPropertyMap	10.0	8.1	4.1
V2	11.0	8.1	4.5?
V3	12.9	<i>none</i>	5.6
AAX_IACFDescriptionHost	12.8	<i>none</i>	<i>none</i>
AAX_IACFFeatureInfo	12.8	<i>none</i>	<i>none</i>

12.55.1.3 Run-Time Interfaces

	Pro Tools	Media Composer	VENUE	
AAX_IACFAutomationDelegate	10.0	8.1	4.1	
AAX_IACFController	10.0	8.1	4.1	
V2	11.0	8.1	none	
V3	12.4	8.6	none	
AAX_IACFEffectDirectData	10.0	8.1	4.1	
V2				
AAX_IACFEffectGUI	10.0	8.1	4.1	
AAX_IACFEffectParameters	10.0	8.1	4.1	
V2	11.0	8.1	4.5?	
V3	11.2	8.1	none	
V4	none	none	5.6	
AAX_IACFHostProcessor	10.0	8.1	none	
V2	12.0	none	none	
AAX_IACFHostProcessorDelegate	10.0	none	none	
V2	11.0	none	none	
V3	12.0	none	none	
AAX_IACFHostServices	10.0	8.1	4.1?	
V2	12.0	8.6	none	
V3	12.8.3	none	none	
AAX_IACFPageTable	12.8	none	5.7	
V2	12.8.2	none	5.7	
AAX_IACFPageTableController	none	none	5.7	
AAX_IACFPrivateDataAccess	10.0	8.1	4.1	
AAX_IACFSessionDocument	2023.6.0	none	none	
AAX_IACFSessionDocumentClient	2023.6.0	none	none	
AAX_IACFTaskAgent	none	none	none	
AAX_IACFTransport	10.0	8.5 (partial)	none	
V2	10.3.7	8.5 (partial)	none	
V3	2021.3		none	
V4	2023.9		none	
AAX_IACFTransportControl	2023.9	none	none	
AAX_IACFViewContainer	10.0	8.1	4.1	
V2	12.0.1	none	none	
V3	2022.4	none	none	

12.55.1.4 Features

	Pro Tools	Media Composer	VENUE	
Basic Stem Formats (mono through 7.1)	10.0	8.1	none	
7.x.2 Stem Formats	12.8	none	none	
Surround Stem Formats (5.0.2 to 9.1.6)	none	none	none	
Ambisonics Stem Formats (first through third order)	12.8.2	none	none	
Ambisonics Stem Formats (fourth through seventh order)	none	none	none	
Plug-in type conversion	10.3.8, 11.1	11.0*, none	none	
Auxiliary Output Stems	10.0	none	none	
Sidechain Inputs	10.0	8.5	none	

MIDI	10.0	<i>none</i>	<i>none</i>	
Automation recording and playback	10.0	<i>none</i>	<i>none</i>	
Plug-in presets	10.0	8.4	4.1	
External control surfaces	10.0	8.1	<i>none</i>	

12.55.2 Host Compatibility Notes

See also

[Compatibility Notes](#) in the [Pro Tools Guide](#) document

Member [AAX_CMidiPacket::mIsImmediate](#)

This value is not currently set. Use `mTimeStamp == 0` to detect immediate packets

Member [AAX_CParameter< T >::AAX_CParameter](#) ([AAX_CParamID](#) identifier, [const AAX_IString &name](#), [T defaultValue](#), [const AAX_ITaperDelegate< T > &taperDelegate](#), [const AAX_IDisplayDelegate< T > &displayDelegate](#), [bool automatable=false](#))

As of Pro Tools 10.2, DAE will check for a matching parameter NAME and not an ID when reading in automation data from a session saved with an AAX plug-ins RTAS/TDM counter part.

As of Pro Tools 11.1, AAE will first try to match ID. If that fails, AAE will fall back to matching by Name.

Module [AAX_DigiTrace_Guide](#)

This feature is available in Pro Tools 12.6 and higher

Member [AAX_eConstraintLocationMask_DLLChipAffinity](#)

This constraint is supported in Pro Tools 10.2 and higher

Member [AAX_eCurveType_Dynamics](#)

Pro Tools requests this curve type for [Dynamics](#) plug-ins only

Member [AAX_eCurveType_EQ](#)

Pro Tools requests this curve type for [EQ](#) plug-ins only

Member [AAX_eCurveType_Reduction](#)

Pro Tools requests this curve type for [Dynamics](#) plug-ins only

Member [AAX_eDataInPortType_Incremental](#)

Supported in Pro Tools 12.5 and higher; when [AAX_eDataInPortType_Incremental](#) is not supported the port will be treated as [AAX_eDataInPortType_Unbuffered](#)

Member [AAX_EHostModeBits](#)

Supported in Venue 5.6 and higher

Member [AAX_eNotificationEvent_ASPreviewState](#)

Supported in Pro Tools 11 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_ASProcessingState](#)

Supported in Pro Tools 11 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_DelayCompensationState](#)

Supported in Pro Tools 12.6 and higher

Member [AAX_eNotificationEvent_EnteringOfflineMode](#)

Supported in Pro Tools 11 and higher

Member [AAX_eNotificationEvent_ExitingOfflineMode](#)

Supported in Pro Tools 11 and higher

Member [AAX_eNotificationEvent_HostModeChanged](#)

Supported in Venue 5.6 and higher

Member [AAX_eNotificationEvent_LogState](#)

Pro Tools currently only sends this notification to the Direct Data object in the plug-in

Member [AAX_eNotificationEvent_MaxViewSizeChanged](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_ParameterNameChanged](#)

Supported in Pro Tools 2023.3 and higher

Member [AAX_eNotificationEvent_PresetOpened](#)

Supported in Pro Tools 11 and higher

Member [AAX_eNotificationEvent_PriorSettingsInvalid](#)

Supported in Venue 5.6 and higher

Member [AAX_eNotificationEvent_SessionBeingOpened](#)

Supported in Pro Tools 11 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_SessionPathChanged](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_SideChainBeingConnected](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_SideChainBeingDisconnected](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_SignalLatencyChanged](#)

Supported in Pro Tools 11.1 and higher

Member [AAX_eNotificationEvent_TrackNameChanged](#)

Supported in Pro Tools 11.2 and higher

Not supported by Media Composer

Member [AAX_eNotificationEvent_TransportStateChanged](#)

Supported in Pro Tools 2021.10 and higher

Member [AAX_ePlugInStrings_Progress](#)

Not currently supported by Pro Tools

Member [AAX_eProcessingState_BeginPassGroup](#)

AudioSuite pass group notifications are supported starting in Pro Tools 12.0

Member [AAX_eProcessingState_EndPassGroup](#)

AudioSuite pass group notifications are supported starting in Pro Tools 12.0

Member [AAX_eProperty_Constraint_NeverUnload](#)

AAX_eProperty_Constraint_NeverUnload is not currently implemented in DAE or AAE

Member [AAX_eProperty_DestinationTrack](#)

This property is not supported on Media Composer

Member [AAX_eProperty_DisableAudiosuiteReverse](#)

AAX_eProperty_DisableAudiosuiteReverse is not currently implemented

Member [AAX_eProperty_LatencyContribution](#)

Maximum delay compensation limits will vary from host to host. If your plug-in exceeds the delay compensation sample limit for a given AAX host then you should note this limitation in your user documentation. Example limits:

- Pro Tools 9 and higher: 16,383 samples at 44.1/48 kHz, 32,767 samples at 88.2/96 kHz, or 65,534 samples at 176.4/192 kHz
- Media Composer 8.1 and higher: 16,383 samples at 44.1/48 kHz, 32,767 samples at 88.2/96 kHz

Member [AAX_eProperty_OptionalAnalysis](#)

In Media Composer, optional analysis will also be performed automatically before each channel is rendered. See [MCDEV-2904](#)

Member [AAX_eProperty_SideChainStemFormat](#)

[AAX_eProperty_SideChainStemFormat](#) is not currently implemented in DAE or AAE

Currently Pro Tools supports only [AAX_eStemFormat_Mono](#) side chain inputs

Member [AAX_eProperty_UsesClientGUI](#)

Currently supported by Pro Tools only

Member [AAX_IACFEffEffectParameters::CompareActiveChunk](#) (const [AAX_SPlugInChunk](#) *iChunkP, [AAX_CBoolean](#) *olsEqual) const =0

In Pro Tools, this method will only be called if a prior call to [GetNumberOfChanges\(\)](#) has indicated that the plug-in's state has changed. If the plug-in's current settings are different from the settings in aChunkP then the plug-in's Compare Light will be illuminated in the plug-in header, allowing users to toggle between the plug-in's custom state and its saved state.

Member [AAX_IACFEffEffectParameters::GetCurveData](#) ([AAX_CTypeID](#) iCurveType, const float *iValues, [uint32_t](#) iNumValues, float *oValues) const =0

Versions of S6 software which support the [GetCurveDataDisplayRange\(\)](#) method will not display a plug-in's curve data unless both [GetCurveData\(\)](#) and [GetCurveDataDisplayRange\(\)](#) are supported by the plug-in.

Member [AAX_IACFEffEffectParameters::GetParameterNameOfLength](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName, [int32_t](#) iNameLength) const =0

In most cases, the AAX host will call [GetParameterName\(\)](#) or [GetParameterNameOfLength\(\)](#) to retrieve parameter names for display. However, when Pro Tools is retrieving a plug-in name for display on a control surface the XML data stored in the plug-in's page tables will be used in preference to values retrieved from these methods.

Member [AAX_IComponentDescriptor::AddAuxOutputStem](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inStemFormat, const char inNameUTF8[]) =0

Pro Tools supports only mono and stereo auxiliary output stem formats

There is a hard limit to the number of outputs that Pro Tools supports for a single plug-in instance. This limit is currently set at 256 channels, which includes all of the plug-in's output channels in addition to the sum total of all of its aux output stem channels.

Member [AAX_IComponentDescriptor::AddClock](#) ([AAX_CFieldIndex](#) inFieldIndex) =0

As of Pro Tools 11.1, this field may be used in both Native and DSP plug-ins. The DSP clock data is a 16-bit cycling counter. This field was only available for Native plug-ins in previous Pro Tools versions.

Member [AAX_IComponentDescriptor::AddMIDINode](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_EMIDINodeType](#) inNodeType, const char inNodeName[], [uint32_t](#) channelMask) =0

Due to current restrictions MIDI data won't be delivered to DSP algorithms, only to AAX Native.

Member [AAX_IController::GetHostName](#) ([AAX_IString](#) *outHostNameString) const =0

Pro Tools versions from Pro Tools 11.0 to Pro Tools 12.3.1 will return a generic version string to this call. This issue is resolved beginning in Pro Tools 12.4.

Member [AAX_IMIDINode::PostMIDIpacket](#) ([AAX_CMidiPacket](#) *packet) =0

Pro Tools supports the following MIDI events from plug-ins:

- NoteOn
- NoteOff
- Pitch bend
- Polyphonic key pressure
- Bank select (controller #0)

- Program change (no bank)
- Channel pressure

Member [AAX_ITransport::GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t *SampleLocation) const =0

There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Member [AAX_ITransport::GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0

This does not indicate anything about the status of the "Loop Record" option. Even when the host is configured to loop playback, looping may not occur if certain conditions are not met (i.e. the length of the selection is too short)

Member [AAX_ITransport::GetCurrentTickPosition](#) (int64_t *TickPosition) const =0

The tick resolution here is different than that of the tick displayed in Pro Tools. "Display ticks" (as they are called) are 1/960 of a quarter note.

Member [AAX_ITransport::GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0

There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Member [AAX_IViewContainer::GetModifiers](#) (uint32_t *outModifiers)=0

Although this method allows plug-ins to acquire the current state of the Windows key (normally blocked by Pro Tools), plug-ins should not use key combinations that require this key.

Module [AAX_Media_Composer_Guide](#)

Some early versions of Media Composer 8 do not search the system plug-ins directory recursively. If your plug-ins are installed into a sub-directory beneath this main directory then they will not be loaded by the affected versions of Media Composer.

Module [AAX_Page_Table_Guide](#)

Pro Tools versions prior to Pro Tools 11.1 use plug-ins' ProControl and ICON page tables (Dynamics, EQ, Channel Strip, Custom Fader, etc.) to map plug-in parameters to EUCON-enabled surfaces, so be sure that your plug-ins also implement these page tables correctly so that users with earlier versions of Pro Tools can have the best possible experience when using your plug-ins.

Module [AAX_Pro_Tools_Guide](#)

Pro Tools requires PACE Eden digital signatures for AAX plug-ins.

Supported in Pro Tools 2019.XX and higher. Also supported (and enabled by default) in Pro Tools developer builds beginning with Pro Tools 2019.6.

Module [AAX_TI_Guide](#)

32 and 64-sample quantum is available in Pro Tools 10.2 and higher

Beginning in Pro Tools 11, AAX DSP algorithms also support optional temporary data spaces that can be described in the Describe module and are shared among all instances on a DSP. This is an alternative to declaring large data blocks on the stack for better memory management and to prevent stack overflows. Please refer to [AAX_IComponentDescriptor::AddTemporaryData\(\)](#) for usage instructions.

Module [AdditionalFeatures_CurveDisplays](#)

For S6 control surface displays, see [PT-226228](#) and [PT-226227](#) in the [Known Issues](#) page for more information about the requirements listed in this section.

Module [advancedTopics_relatedTypes](#)

Pro Tools versions prior to Pro Tools 12.3 do not allow explicit type conversion between types with different product ID values. Beginning in Pro Tools 12.3 both the product ID and the plug-in ID may differ between explicitly related types.

Module [AuxInterface_TaskAgent](#)

This interface is not yet used in any AAX hosts

Module [CommonInterface_Algorithm](#)

As of Pro Tools 10.2.1 an algorithm's initialization callback routine will have up to 5 seconds to execute.

Module [CommonInterface_FormatSpecification](#)

*_ACFGetSDKVersion is required for 64-bit AAX plug-ins only

Module [ExamplePlugins](#)

The DemoDelay_DynamicLatencyComp example is compatible with Pro Tools 11.1 and higher.

Collaboration diagram for Host Support:



12.56 Known Issues

A list of known bugs affecting AAX plug-ins.

12.56.1 Contents

- [Known Issues in the AAX SDK](#)
- [Known Issues in Pro Tools](#)
- [Known Issues in Venue Live Sound Systems](#)
- [Known Issues in Media Composer](#)
- [Known Issues in Control Surfaces](#)
- [Known Issues in Other Software](#)
- [Known Issues in AAX Tools](#)

12.56.2 Known Issues in the AAX SDK

12.56.2.1 AAXSDK-897

Calling [AAX_CParameter](#) SetValue() methods during [AAX_CEffectParameters::EffectInit\(\)](#) does not set the parameter value

As a workaround, use [AAX_CParameter::SetNormalizedDefaultValue\(\)](#) to control the initial value of a parameter.

Resolution: This bug is unresolved

12.56.2.2 AAXSDK-851

The `register` keyword is incorrectly used in [AAX_Atomic.h](#)

This causes compilation failures in clang on Windows

Resolution: This bug is unresolved

12.56.2.3 AAXSDK-832

Rectifi example does not output any signal in DSP mode

Details: The AAX SDK example version of Rectifi initiates but does not output any sound when in DSP mode

Resolution: This bug is unresolved

12.56.2.4 AAXSDK-708

The AAX SDK library will not compile using macOS SDK 10.13

Details: Compilation results in an error in NSUUID.h. This error does not occur when using SDK 10.14 or later

Resolution: This bug has been fixed in the macOS SDK

12.56.2.5 AAXSDK-705

[AAX_VHostProcessorDelegate](#) does not detect hosts with [V2](#) support

Resolution: This bug is fixed as of AAX SDK 2.4.0

12.56.2.6 AAXSDK-663

AAX SDK `#pragma pack` errors with XCode 10 and later

Resolution: This bug is fixed as of AAX SDK 2.3.2

12.56.2.7 AAXSDK-599

In the Win32 GUI example plug-in, the mouse cursor disappears when text is entered in text box and only re-appears when the mouse is moved out of the plug-in window bounds

Resolution: This bug is unresolved

12.56.2.8 AAXSDK-561 / PT-232159

An AAX Hybrid DSP plug-in with 7.1.2 input and output stem formats and a 16-sample processing buffer size will throw an AAE -14382 error upon instantiation at 192kHz

Resolution: This bug is unresolved

12.56.2.9 AAXSDK-533

AAXLibrary compiles with warnings in Visual Studio 2015

Resolution: This bug is fixed as of AAX SDK 2.3.0

12.56.2.10 AAXSDK-514

Using collection-level properties leads to a leaked ACF object

Resolution: This bug is fixed as of AAX SDK 2.3.0

12.56.2.11 AAXSDK-321

Demo Delay (mono) DSP / Demo Gain (Cocoa UI) (mono) DSP can't be instantiated

Resolution: This bug is fixed as of AAX SDK 2.2.0

12.56.2.12 AAXSDK-271

DemoMIDI_Sampler: No audio on right channel (multi-mono)

Resolution: This bug is fixed as of AAX SDK 2.2.0

Discussion: DemoMIDI_Sampler and DemoMIDI_Synth are now restricted to not use multi-mono, via the [AAX_eProperty_Constraint_MultiMonoSupport](#) property.

In Pro Tools, when a track's MIDI destination is set to "none" and a new plug-in that includes a MIDI input node (e.g. any Instrument plug-in) is instantiated, the track's MIDI destination is set to the first newly created MIDI input node.

When the track in question is greater than mono, and the Instrument plug-in is multi-mono, the default behavior is for the track's MIDI destination to be set to the first of the newly created input nodes, not to all of them simultaneously. As a result, the track's MIDI destination is set to the MIDI input node of the first (left) multi-mono instance of the plug-in.

To route MIDI to all channels of a multi-mono plug-in in Pro Tools, ctrl-select the MIDI destination and choose the additional MIDI nodes.

12.56.2.13 AAXSDK-186

C99Compatibility constructs are incorrect when used with VS2012

Resolution: This bug is fixed as of AAX SDK 2.1.1

12.56.2.14 AAXSDK-162

Misleading warning message when attempting hardware debugging using a non-local TIShell.out

The "Load ProTools Plug-in Symbols" script gives the following warning: D:/Code_7/dev.ws.↔ backup-win7-concert/AAX/Internal/SystemSoftware/TIShell/CCS_Project/TIShell/../../../../WinBag/x64/Release/bin/TIShell.out does not exist! Please question everything.

This error message includes a hard-coded path that is not relevant to the running system. This error is benign but it can be confusing.

Resolution: This bug is unresolved

12.56.2.15 AAXSDK-16

AAX SDK: Win32 example plug-in GUI does not appear in Windows 8

Resolution: This bug is fixed as of AAX SDK 2.2.0

12.56.2.16 AAXSDK-14

AAX DemoGain_VST: Text box entry is not acknowledged upon click outside of window

Resolution: This bug is unresolved

12.56.2.17 AAXSDK-13 / AAX-579 / PTSW-158381

AAX SDK Win32 example plug-in does not "snap to default" on option-click

Resolution: This bug is fixed as of AAX SDK 2.2.0

12.56.2.18 AAXSDK-11 / AAX-581 / PTSW-158348

AAX SDK VSTGUI example plug-in does not respond to 'alt' or 'win' modifier keys (Windows)

Resolution: This bug is fixed as of AAX SDK 2.2.0

12.56.2.19 AAXSDK-10 / AAX-580 / PTSW-154083

AAX DemoGain_VST and DemoGain_Cocoa require initial click on GUI to take focus before editing (macOS)

Resolution: This bug is not yet resolved. For macOS, one workaround is to modify VSTGUI's cocoasupport.mm file in order to add a handler for the acceptsFirstMouse selector:

```
static BOOL VSTGUI_NSView_acceptsFirstMouse(
    id self,
    SEL _cmd)
{
    return YES;
}

// In VSTGUI_NSView_isOpaque()
res = class_addMethod(
    viewClass,
    @selector(acceptsFirstMouse:),
    IMP (VSTGUI_NSView_acceptsFirstMouse),
    "B@:@");
```

Special thanks to Nick Protokowicz for suggesting this workaround.

12.56.2.20 AAXSDK-6 / AAX-646

AAX SDK: Incorrect output from scatter/gather DMA example plug-in when increasing playback buffer size while audio is present (Native decks)

Resolution: This bug is unresolved

Workaround: The workaround for this issue is to not run audio through this plug-in while increasing the playback buffer size.

12.56.2.21 AAXSDK-5

[AAX_CChunkDataParser::LoadChunk](#) doesn't handle unknown chunk items well

Resolution: This bug will not be fixed

Discussion: [AAX_CChunkDataParser](#) does not store the size of each chunk item in the data stream. Therefore there is no way to determine the correct size for each data element when reading a chunk that was generated by this parser.

Workaround: If you know the correct size of each data element in a chunk when it is read by the plug-in, you can override the [AAX_CChunkDataParser](#) methods to ensure that each data element is correctly sized.

12.56.2.22 AAXSDK-2 / AAX-648

AAX SDK: Output from DMA example plug-in is one buffer early

Resolution: This bug is unresolved

12.56.2.23 AAX-582 / PTSW-157726

AAX SDK example plug-ins' controls do not write automation properly when in 'touch' mode (frequently revert to default value while writing)

Resolution: This bug is fixed as of the 1.0.4 SDK

12.56.2.24 AAX-585 / PTSW-157451

AAX DemoGain GUI example plug-ins do not correctly handle alt/opt-click for resetting controls to their default state

Resolution: This bug is partially resolved as of AAX SDK 1.0.4. See also PTSW-158348 and PTSW-158381.

12.56.2.25 AAX-578 / PTSW-158310

AAX SDK JUCE example plug-in does not "snap to default" on option-click

Resolution: This bug is fixed as of AAX SDK 1.0.4

12.56.3 Known Issues in Pro Tools

subsection PT-307986 Certain Avid plugins, such as SynthCELL and GrooveCELL, prevent AAX plugins from forking subprocesses

Resolution: This bug is unresolved

12.56.3.1 PT-307193

Committing creates a consolidated clip when plug-in sets AAX_eProperty_Constraint_AlwaysProcess to true

Resolution: This bug is unresolved

12.56.3.2 PT-305352

Automation values change unpredictably when changing a plug-in parameter from continuous to discrete or discrete to continuous

Resolution: This bug is unresolved

12.56.3.3 PT-303482

(Important Escalation) Contextual menus and child windows miss mouse clicks in some 3P plug-ins, new to PT 2023.3

Resolution: This bug is Closed as Fixed in PT 2023.R2.0

12.56.3.4 PT-299906

AudioSuite plug-ins with sidechains crash PT (arm64 only)

Resolution: This bug is Closed as Fixed in PT 2023.R1.0

12.56.3.5 PT-297802

Integer overflow in [AAX_ITransport::GetTimelineSelectionStartPosition](#) on Windows

[AAX_ITransport::GetTimelineSelectionStartPosition\(\)](#) supplies an incorrect sample position when the value is greater than about 12.42 hours at 48kHz.

Resolution: This bug is fixed in Pro Tools 2023.9

12.56.3.6 PT-290588

AudioSuite offline processing option cannot be used after performing certain actions (it simply becomes greyed out)

Resolution: This bug is fixed as of Pro Tools 2022.12

12.56.3.7 PT-284916

Modifications to non-automatable plug-in parameters do not set the Pro Tools session dirty flag for AAX plug-ins that disable default settings via [AAX_eProperty_Constraint_DoNotApplyDefaultSettings](#)

Resolution: This bug is unresolved

12.56.3.8 PT-282946

AAX timestamps for live plug-ins are two buffers out of sync with timestamps for non-live plug-ins

This relates to timestamps derived from [AAX_IComponentDescriptor::AddClock\(\)](#)

Resolution: This bug is unresolved

12.56.3.9 PT-278282

Crash when running certain Accelerate.framework operations

Resolution: This bug is fixed as of Pro Tools 2021.12

12.56.3.10 PT-276280

The VoiceOver accessibility tree for JUCE based plug-in GUIs is not connected to the Pro Tools plug-in window

Resolution: This bug is unresolved

12.56.3.11 PT-274717

AudioSuite settings are saved with the system rather than with the session

Resolution: This bug is unresolved

12.56.3.12 PT-271830

Crash when re-sizing a window for a plug-in that uses JUCE with OpenGL rendering enabled

Details: When Pro Tools resizes an AAX plug-in window it can cause the the `GL rendercontext` used by JUCE to become invalidated. Plug-ins must take care not to use this object within the scope of a concurrent window re-size request via [AAX_IViewContainer::SetViewSize\(\)](#) .

Resolution: Plug-in developers must synchronize access to any OpenGL objects that are not thread-safe.

12.56.3.13 PT-263909

Clipboard is pasted twice when pasting text into a JUCE plug-in text box

Resolution: This bug is fixed as of Pro Tools 2022.12

12.56.3.14 PT-263859

Committing up to an insert that is followed by a width-converting insert also commits the width-converting insert

Resolution: This bug is unresolved

12.56.3.15 PT-261394

Frame rate offsets are calculated incorrectly for plug-ins when the session is at a higher frame rate

Resolution: This bug is fixed as of Pro Tools 2020.5

12.56.3.16 PT-258560 / PT-256919

Multi-input only AudioSuite plug-ins are processed as multi-mono

Details: Plug-ins that define [AAX_eProperty_MultiInputModeOnly](#) actually get mono mode only

Resolution: This bug is unresolved

12.56.3.17 PT-258394

JUCE [AAX](#) plug-ins which use images from `BinaryData` crash on Catalina

Resolution: This bug is unresolved

12.56.3.18 PT-257213

AAX Hybrid plug-ins produce distorted signal on tracks in DSP Mode when using the HDX Hybrid Engine or Pro Tools Carbon

Details: Because of this bug, we have disabled DSP Mode for certain AAX Hybrid plug-ins.

Resolution: The fix is ending bug review

12.56.3.19 PT-256704

Pro Tools Plug-In Folder permissions (macOS) are set to allow write access by anyone

Resolution: This bug is fixed as of Pro Tools 2019.12

12.56.3.20 PT-255800

AAX Hybrid plug-in output on HDX contains a gap which varies with host buffer size

Resolution: This bug is unresolved

12.56.3.21 PT-255408

Changing one plug-in insert results in a redundant audio buffer on a later insert in the chain

Resolution: This bug from an external report could not be verified by Avid

12.56.3.22 PT-254203

Reverb and Delay AudioSuite plug-ins cannot provide an "Analysis" button

Resolution: This bug is unresolved; the [AAX_eProperty_DisableAudiosuiteReverse](#) property has not yet been implemented.

12.56.3.23 PT-254118 / PT-275441 / PT-279941

Dynamic Plug-In Processing doesn't work during playback

Details: Plug-ins are processed continuously during playback even when Dynamic Plug-In Processing is engaged. This issue has been fixed for certain track and plug-in types in various recent releases, but there are still common configurations for which dynamic plug-in processing does not work.

Resolution: This bug is fixed as of Pro Tools PT 2023.3

12.56.3.24 PT-254103

Some AAX plug-ins are displayed with incorrect colors

Details: This issue affects plug-ins that use OpenGL in Pro Tools 2019.5 and higher. It relates to a Pro Tools optimization in which color space conversion is skipped for certain parts of the application UI.

Resolution: This bug is unresolved

12.56.3.25 PT-250751

AudioSuite plug-ins do not set a [custom suffix](#) on a clip in case the selection reference is "Clips list"

Resolution: This bug is unresolved

12.56.3.26 PT-249791

AudioSuite: [Custom clip name](#) is not applied to generated audio file on disk

Resolution: This bug is unresolved

12.56.3.27 PT-249790

AudioSuite plug-ins cannot set a [custom clip name suffix](#)

Resolution: This bug is fixed as of Pro Tools 2019.10

12.56.3.28 PT-248000

Plug-in partial bypass should support more than just EQ and Dynamics categories

Resolution: This enhancement is not yet supported

12.56.3.29 PT-245693

Dynamic plug-in processing on HDX cuts off reverb tails

Resolution: This bug is unresolved

12.56.3.30 PT-243211

Crash when closing a plug-in window unless the plug-in leaks its [AAX_IViewContainer](#) reference counts

Resolution: This bug is fixed as of Pro Tools 2019.5

12.56.3.31 PT-237857

Custom EQ curve display ranges are not supported

Resolution: This enhancement is not yet supported

12.56.3.32 PT-236755

Up-mixing plug-ins with AOS drop output channels on HDX

Resolution: This bug is fixed as of Pro Tools 2018.7

12.56.3.33 PT-235831

The plug-in frame overlaps the plug-in window header if the plug-in GUI is taller than the screen height less the plug-in window header height.

Resolution: This bug is unresolved

Workaround: Avoid resizing the plug-in GUI to a height greater than the display height.

12.56.3.34 PT-235333

Dynamic plug-in processing incorrectly shuts off processing in mixed multi-mono/multichannel chains that should force processing

This bug can occur if a multi-mono plug-in preceeds a multichannel plug-in which sets [AAX_eProperty_Constraint_AlwaysProcess](#)

Resolution: This bug is unresolved

12.56.3.35 PT-234681

AAX plug-in parameter handling may cause audio glitches on Windows for plug-ins with very long [GenerateCoefficients\(\)](#) execution time

Resolution: This bug is unresolved

12.56.3.36 PT-233726

Unprintable characters in four-char parameter IDs may result in -9105 errors

Resolution: This bug is fixed as of Pro Tools 2018.1

12.56.3.37 PT-233176

AAX digital signature check fails on pre-Sierra systems for plug-ins signed on Sierra

Resolution: This bug has been reported to Avid but is not yet confirmed. Contact PACE Anti-Piracy, Inc. if you encounter this behavior.

12.56.3.38 PT-232678 / PT-236755

Plug-in aux outputs are silent for upmix plug-ins when using AAX Native plug-ins with the HDX playback engine

Resolution: This bug is fixed as of Pro Tools 2018.1

12.56.3.39 PT-232403

ProTools shows error if the plug-in's multi-chunk preset file contain incorrect chunk listed in the end

For example, if a new chunk type has been added to a later version of a plug-in and a preset from that version is opened in an earlier version of the plug-in.

Resolution: This bug is fixed as of Pro Tools 12.8.2

12.56.3.40 PT-232159

AAX Hybrid plug-ins with more than 16 total input channels (direct input and hybrid input) raise AAE -14382 error upon instantiation at 192 kHz sample rate

Resolution: This bug is unresolved

12.56.3.41 PT-230327

The AAX related types feature is broken ([AAX_IACFPropertyMap_V3](#) inheritance is incorrect)

Resolution: This bug was introduced in Pro Tools 12.8 and is fixed as of Pro Tools 12.8.1

12.56.3.42 PT-230290

The plug-in preset menu takes a long time to build for plug-ins with a very large number of preset .tfx files

Resolution: This bug is fixed as of Pro Tools 2018.7

12.56.3.43 PT-230288

The plug-in preset menu contains empty folders for other Effects in the same .aaxplugin bundle

Resolution: This bug is fixed as of Pro Tools 12.8.2

12.56.3.44 PT-229026

Pro Tools may crash after plug-in parameter tweaks on Windows

Resolution: This crash has not been reproduced starting with Pro Tools 2018.1

12.56.3.45 PT-227655

[AAX_ITransport::GetTimelineSelectionStartPosition\(\)](#) provides incorrect values for real-time plug-ins - value depends on transport time display selection in Pro Tools

Resolution: This bug is fixed as of Pro Tools 12.8

12.56.3.46 PT-227173

Incorrect timecode is sent to plug-ins when delay is present before the plug-in

Resolution: This bug is unresolved

Workaround: Attach a debugger after opening a session in Pro Tools. After the first session open, subsequent session open actions will not result in a crash.

12.56.3.47 PT-226559

Transport location provided to plug-ins is incorrect during half-speed playback

Resolution: This bug is unresolved

12.56.3.48 PT-225763

Incorrect AudioSuite processing modes are available for multichannel random access plug-ins

Resolution: This bug is unresolved

12.56.3.49 PT-223581

Pro Tools removes plug-ins from the insert menu if unsupported stem formats are detected

Resolution: This bug is fixed as of Pro Tools 12.8

This bug is also now fixed in earlier versions of Pro Tools via a workaround which is now built into the AAX SDK during Describe. See [AAX_VPropertyMap::AddProperty\(\)](#)

12.56.3.50 PT-218545

There is no way for AAX plug-ins to opt out of the default settings chunk sequence during plug-in instantiation

Resolution: This enhancement will be supported in a Pro Tools 2019 release; see [AAX_eProperty_Constraint_DoNotApplyDefaultSettings](#)

12.56.3.51 PT-218486

Removing all automation parameters doesn't stop control from jumping back on first playback

Resolution: This bug is unresolved

12.56.3.52 PT-210904 / VSW-14216

HDX errors can occur due to over-allocation of certain plug-ins

This bug applies specifically to AAX DSP plug-ins which register the same DLL and algorithm entry point name for multiple modules with different [AAX_eProperty_TI_MaxInstancesPerChip](#) requirements.

In VENUE, this bug causes a 'No Information Available' error dialog on a single DSP chip

Resolution: This bug is fixed as of Pro Tools 12.5 and VENUE 5.1

12.56.3.53 PT-206995

AOS is not cleaned up in [AAX_IComponentDescriptor::Clear](#)

Resolution: This bug will not be fixed

12.56.3.54 PT-206541

AAX automation playback is late and non-deterministic

Resolution: This bug is unresolved

12.56.3.55 PT-206161

HDX: AAX packets are not delivered to ports 16 or 24 (zero-indexed) when `PostPacket()` is called outside of `GenerateCoefficients()`

Workaround: Make all calls to `AAX_IController::PostPacket()` within the scope of `AAX_IEffectParameters::GenerateCoefficients()`

Resolution: This bug will not be fixed

12.56.3.56 PT-205610

AAX Hybrid: transport location and clock methods do not provide correct values when called from the Hybrid render callback

Resolution: This bug is fixed as of Pro Tools 12.4

12.56.3.57 PT-203420

`TestGetCurveData` DigiOption results in incorrect plug-in view offset for plug-ins with MIDI

Resolution: This bug will not be fixed

Workaround: Temporarily disable MIDI in your plug-in while developing or debugging the plug-in's curve data, then re-enable MIDI once you have finished using the `TestGetCurveData` DigiOption.

12.56.3.58 PT-202345

Pro Tools may incorrectly identify plug-ins when a single plug-in uses identical plug-in IDs across different Effects

Resolution: This bug is fixed as of Pro Tools 12.2

12.56.3.59 PTSW-200437 / PTSW-197598

Plug-Ins that use the "Related Types" feature cannot relate to plug-ins with a different ProductID

Resolution: This bug is fixed as of Pro Tools 11.3.2 and Pro Tools 10.3.11

12.56.3.60 PTSW-197651 / PT-218405

In some cases AAX plug-ins do not show the correct Control Name Variation and just read out the automation name

Resolution: This bug is will not be fixed

12.56.3.61 PTSW-197601 / PT-218459

Control surfaces can send illegal parameter values to plug-ins

Resolution: This bug will not be fixed

12.56.3.62 PTSW-197593 / PT-218480

HDX: A chip may be full with just a small percent of the System Usage meter filled

Resolution: This bug will not be fixed

12.56.3.63 PTSW-197540

AudioSuite: Analyze mode is not working properly when processing method is set to "clip-by-clip"

Resolution: This bug is fixed as of Pro Tools 11.3.1

12.56.3.64 PTSW-197472

Dynamic Plug-In Processing is unnecessarily disabled for plug-ins in the "Other" category

Resolution: This bug is fixed as of Pro Tools 12

12.56.3.65 PTSW-197471

AudioSuite only analyzes the first clip in "clip list" mode

Resolution: This bug is fixed as of Pro Tools 12

12.56.3.66 PTSW-197468 / PT-218460

Pro Tools may incorrectly change a plug-in instance when another instance is edited

Resolution: This bug is fixed in Pro Tools 2018.1

Discussion: This issue only happens when the two plug-in types use the same Manufacturer and Plug-In IDs and use Product IDs with unprintable chars when interpreted as four-char values.

We strongly recommend that you select Product IDs in the printable ASCII four-char range.

12.56.3.67 PTSW-197431 / PT-218414

Pro Tools 10: Plug-in Side-chain input is silent when the plug-in supports Aux Outputs

Resolution: This bug will not be fixed

12.56.3.68 PTSW-197075

Plug-in preset files for some plug-ins are not cross-compatible between Mac and Windows

Resolution: This bug is fixed as of Pro Tools 11.2 and Pro Tools 10.3.10

12.56.3.69 PTSW-196772 / PT-218423

A [View Size Changed](#) notification is not sent when connecting/disconnecting a display

Resolution: This bug will not be fixed

12.56.3.70 PTSW-196604

Cannot adjust plug-in parameters with large numbers of steps using control surface (Artist Series)

Resolution: This bug is fixed as of Pro Tools 12.4

12.56.3.71 PTSW-196428 / PT-218488

AudioSuite preview allows processing with incorrect number of channels

Resolution: This bug will not be fixed

12.56.3.72 PTSW-195316 / PT-218485

EQ/Dyn graphs on EUCON surfaces are not always updated for plug-ins with many EQ/Dyn parameters

Resolution: This bug will not be fixed

Discussion: Currently, EUCON surfaces only update a plug-in's EQ/Dyn curve plots when an update occurs to one of the parameters which is mapped to the plug-in's "center section" EQ/Dyn page tables. Other parameters will not trigger an update to the plug-in's EQ/Dyn curve plot.

12.56.3.73 PTSW-195257

Calling [AAX_ICollection::AddEffect\(\)](#) multiple times using the same `iEffectID` only returns an error if called with the same [effect descriptor](#), but this is always illegal

Resolution: This bug is fixed as of Pro Tools 12

Discussion: Calling [AAX_ICollection::AddEffect\(\)](#) using the same ID will now return an error in all cases

12.56.3.74 PTSW-195256 / PT-218429

Plug-in is not notified of preset load when loading factory default presets

Resolution: This bug will not be fixed

12.56.3.75 PTSW-195209 / PT-218474

[AAX_IViewContainer::HandleParameterMouseUp\(\)](#) returns [AAX_ERROR_UNIMPLEMENTED](#) when using control-command-option-click on a plug-in GUI control

Resolution: This bug will not be fixed

Discussion: See the discussion of this bug in the [AAX_IViewContainer](#) documentation.

12.56.3.76 PTSW-195113

Automation problems with plug-in parameter names > 31 characters

Resolution: This bug is fixed as of Pro Tools 11.2.1

Discussion: This bug was introduced in Pro Tools 11.1

12.56.3.77 PTSW-194698 / PT-218478

Very hard to edit plug-in parameters with many steps using a control surface rotary encoder

Resolution: This bug will not be fixed

12.56.3.78 PTSW-194231 / PT-218434

When the output on a track is set to "no output" then no audio is sent to Auxiliary Outputs of the plug-ins on the track

Resolution: This bug is unresolved

Workaround: Users can work around this issue by ensuring that a track output is always assigned for tracks with plug-ins that generate Auxiliary Output channels. For example, the user may pull down the track's output gain fader, enable MUTE, or select an unused output channel or bus.

12.56.3.79 PTSW-193646

AudioSuite plug-ins are not able to partially re-name clips

Resolution: This bug is fixed as of Pro Tools 12

12.56.3.80 PTSW-193400

AOS plug-ins become active when moving an inactive plug-in to another insert

Resolution: This bug is fixed as of Pro Tools 11.2

12.56.3.81 PTSW-193345

AudioSuite processing notifications are not sent at the start and end of a processing event

Resolution: This bug is fixed as of Pro Tools 12

Discussion: Two new notifications were added to provide this behavior. The existing AudioSuite notifications retain their behavior: they are sent before and after each processing pass, i.e. at the beginning and end of each audio channel that is processed, even if the current selection includes multiple channels. For more information, see [AAX_EProcessingState](#)

12.56.3.82 PTSW-193339

Pro Tools does not update plug-in settings when a new setting's name matches an old setting and the modification date is later

Resolution: This bug will not be fixed

Workaround: To update the settings that are bundled with a plug-in, the plug-in's installer should search for and remove any deprecated settings files on the system.

12.56.3.83 PTSW-193051

Using Aux Output Stems on DSP plug-ins causes them to crash

Resolution: This bug is fixed as of Pro Tools 11.2

Discussion: This bug was introduced in Pro Tools 11.1

12.56.3.84 PTSW-192863 / PT-218498

Plug-in side chain input is not properly delay compensated: aligned with output instead of input, no individual tap per insert

Resolution: This bug is fixed as of Pro Tools 2021.6

12.56.3.85 PTSW-192755

Key focus is not returned to a plug-in after it launches a dialog

Resolution: This bug will not be fixed (design limitation)

12.56.3.86 PTSW-192720 / PT-218467

External source (SideChain) key input is not reported to DSP Dynamics plug-ins after HDX re-shuffle, hence no Gain Reduction occurs.

Resolution: This bug is unresolved

12.56.3.87 PTSW-192635

Implement Manufacturer ID byteswap

Resolution: This bug is fixed as of Pro Tools 11.2 and Pro Tools 10.3.10

12.56.3.88 PTSW-192456 / PT-218490

Plug-in settings chunks with incorrect fSize result in junk data

Resolution: This bug will not be fixed

12.56.3.89 PTSW-192251 / PT-218394

EUCON surface cells sometimes behave inconsistently when discrete plug-in parameters are mapped to rotary encoders

Resolution: This bug will not be fixed

12.56.3.90 PTSW-192086 / PT-218465

AudioSuite AAX: Pro Tools performs multiple unnecessary render passes when rendering in multi-input mode

Resolution: This bug will not be fixed

12.56.3.91 PTSW-191875

Pro Tools uses a hard-coded version string when publishing its version to AAX plug-ins ([AAX_IController::GetHostName](#))

Resolution: This bug is fixed as of Pro Tools 12.4

12.56.3.92 PTSW-191446 / PT-218600

Global symbols due to statically linked boost libs in Pro Tools components may conflict with plug-in components that use boost

Resolution: This bug is unresolved

12.56.3.93 PTSW-191317 / PT-218425

The Pro Tools meter decay setting is not applied to plug-in meters

Resolution: This bug will not be fixed

12.56.3.94 PTSW-191139

Plug-ins do not receive parameter touch state when automation-enabled in Pro Tools 11.1

Resolution: This bug is fixed as of Pro Tools 11.1.2

12.56.3.95 PTSW-190722

Some plug-in state changes do not trigger the Pro Tools session "dirty" flag

Resolution: This bug is fixed as of Pro Tool 11.1.2 and and Pro Tools 10.3.9

12.56.3.96 PTSW-190719

Unexpected behavior for plug-in auxiliary output channels > 128

Resolution: This bug will be fixed as of Pro Tools 11.1.3

12.56.3.97 PTSW-190340

In some cases AAX plug-ins do not show the correct Control Name Variation on control surfaces

This bug can occur when a page table references parameters by ID and some parameters' ID strings are exactly as long as the control surface's display. In this scenario, the control surface will display the parameter's ID string rather than a Control Name Variations (abbreviation) string of equivalent length.

Resolution: This bug is fixed as of Pro Tools 12

12.56.3.98 PTSW-189928 / PT-218456

Failure to load AAX plug-ins with spaces in DLL filename

Resolution: This bug will not be fixed

12.56.3.99 PTSW-189738 / PT-218494

AAX: [AAX_IController::PostPacket\(\)](#) doesn't return any error if you attempt to post to a private data field

Resolution: This bug will not be fixed

12.56.3.100 PTSW-189725 / PT-218397

Auto-generated AudioSuite plug-in GUIs are non-functional for the first plug-in loaded into the window

This bug applies to AudioSuite plug-ins which use the [AAX_eProperty_UsesClientGUI](#) property. This bug is present in all Pro Tools versions which support AAX.

Workaround: This bug only applies when an AudioSuite window is first created. To resolve the issue, toggle an open AudioSuite window between different plug-ins. After toggling to another plug-in and back to the original plug-in, the auto-generated GUI will again be functional.

Resolution: This bug will not be fixed

12.56.3.101 PTSW-189439 / PT-218427

Attempts to set signal latency by non-linear AudioSuite plug-ins should fail, but do not

Resolution: This bug will not be fixed

12.56.3.102 PTSW-189279

An AudioSuite plug-in ID may be incorrectly used as a related type, preventing type-swapping

Resolution: This bug is fixed as of Pro Tools 11.2 and Pro Tools 10.3.10.

12.56.3.103 PTSW-188836 / PT-218428

[AAX_CMidiPacket::mIsImmediate](#) field is not getting set for real-time MIDI messages

Resolution: This bug will not be fixed

12.56.3.104 PTSW-188830

AudioSuite: [PreRender\(\)](#) is not called before each preview pass

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.105 PTSW-188653 / PT-218451

Plug-ins are not unloaded and cannot be swapped when EnablePlugInHotSwap option is enabled

Resolution: This bug will not be fixed

The workaround for this issue is to re-launch Pro Tools after installing a new build of the plug-in

12.56.3.106 PTSW-188161

It is not possible to launch some Pro Tools 11 and later development builds from a debugger

Resolution: This bug is unresolved. It applies to all Pro Tools 11 and higher development builds on Windows and to some development builds on Mac.

Workaround Use the `PauseDuringLaunchToAttachDebugger` [DigiOption](#) to attach the debugger at a safe point in the Pro Tools launch process

12.56.3.107 PTSW-187670

Plug-in preset menu takes a long time to load with many presets

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.108 PTSW-187220 / PT-218584

[AAX_ePrivateDataOptions_KeepOnReset](#) is not implemented

Resolution: This bug is unresolved

12.56.3.109 PTSW-187216 / PT-218491

Pro Tools has a problem with [AAX_IController::PostPacket\(\)](#) being called during [AAX_IEffectParameters::TimerWakeup\(\)](#)

Resolution: This bug will not be fixed

Workaround: This bug occurs only with unbuffered plug-ins' ports for coefficients, so the workaround for this issue is to use buffered ports instead.

12.56.3.110 PTSW-187159

Plug-in parameters can get stuck in touched state; touch/release tokens do not always match

One race condition that could result in this bug behavior has been addressed in Pro Tools 11.1.0. However, the bug can still occur when using EUCON control surfaces due to a mismatch in touch/release tokens sent from those surfaces.

Resolution: This bug is resolved as of Pro Tools 11.1.3. It is unresolved in Pro Tools 10

12.56.3.111 PTSW-187066 / PT-218391

[AAX_ITransport::GetCurrentNativeSampleLocation\(\)](#) returns invalid value on the start of playback

Resolution: This bug will not be fixed

12.56.3.112 PTSW-186864

Automatable parameter values may change between [Set](#) and [Update](#)

Resolution: This bug is unresolved

12.56.3.113 PTSW-186725

Related types do not work when used with [AAX_eProperty_SampleRate](#)

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.114 PTSW-186627

AAX plug-ins whose context field IDs are not defined in Describe cause a crash in Pro Tools

Resolution: This bug is closed (unable to reproduce)

12.56.3.115 PTSW-186253

AudioSuite GUI work causes audio playback glitches and stutters

Resolution: This bug is fixed as of Pro Tools 11.2.

12.56.3.116 PTSW-186189

If an AAX plug-in does not declare all the fields in its context block, undefined behavior may occur (possibly a crash)

Resolution: This bug is fixed as of Pro Tools 10.3.8 and Pro Tools 11.0.2

12.56.3.117 PTSW-186182

On Windows, VSTGUIv4 plug-in GUIs do not receive key events (PT11 only)

Resolution: This bug is addressed with a patch to the AAX SDK's VSTGUI extension implementation as of AAX SDK 2.1.0

12.56.3.118 PTSW-185868 / PT-218439

AAX: Calls to [SetValue\(\)](#) early in plug-in life may not propagate to [UpdateParameterNormalizedValue\(\)](#)

Resolution: This bug will not be fixed

The workaround for this issue is to call SetValue redundantly until the desired value is updated.

12.56.3.119 PTSW-185867 / PT-218470

Session tempo should be available during [EffectInit\(\)](#)

Resolution: This bug will not be fixed

Workaround: The workaround for this issue is to poll the transport interface in [TimerWakeup\(\)](#) or otherwise call it after [EffectInit\(\)](#) completes.

12.56.3.120 PTSW-185866

Pro Tools does not respond to [SetParameterNormalizedValue\(\)](#) while offline bouncing

Resolution: This bug is fixed as of Pro Tools 11.1. However, note that we do not recommend implementing linked parameters using direct calls to [SetParameterNormalizedValue\(\)](#). For an explanation of the correct approach to parameter linking, see [Linked parameters](#), with examples provided in the SDK example plug-ins.

12.56.3.121 PTSW-185825 / PT-218464

Undo key events do not reach plug-ins (Windows)

Resolution: This bug will not be fixed

12.56.3.122 PTSW-185537

Use of DigiTrace results in eTISysSwapScriptTimeout

Resolution: This bug fixed as of Pro Tools 11.1

12.56.3.123 PTSW-185484

[AAX_TRACE_RELEASE](#) crashes at highest optimization setting in AAX DSP plug-ins

Resolution: This bug is unresolved

12.56.3.124 PTSW-185483

DigiTrace: Only one parameter can be sent per trace on HDX

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.125 PTSW-185462

AudioSuite: Error 1224 on AudioSuite render when significantly changing the length of a clip (Windows 8)

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.126 PTSW-185343

[AAX_ITransport::GetTimeCodeInfo](#) returns invalid values for AAX Instruments

Resolution: This bug is fixed as of Pro Tools 10.3.7 and Pro Tools 11.0.2

12.56.3.127 PTSW-185341

Related types come up as inactive when going from HDX > Native

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.128 PTSW-184777 / PT-218483

AAX plug-in meters are not cleared during silence

This bug is new to Pro Tools 11. It does not occur in Pro Tools 10.

Resolution: This bug will not be fixed

12.56.3.129 PTSW-184770

AAX Hybrid plug-ins cannot be opened as AudioSuite (AAE -7103 error)

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.130 PTSW-184682

Incorrect audio buffer length provided when a native plug-in (erroneously) registers [AAX_eProperty_AudioBufferLength](#)

Resolution: Since this is an unsupported plug-in configuration this bug will not be fixed

12.56.3.131 PTSW-184642 / PT-218627

AudioSuite "progress" dialog re-naming is not supported by AAX (it was supported in Pro Tools 9 and earlier)

Resolution: This feature is not implemented

12.56.3.132 PTSW-184619 / PT-218473 / AAX-600

AAX MIDI plug-ins' MIDI channels are not uniquely labeled

Resolution: This bug will not be fixed

12.56.3.133 PTSW-184541

Native engine strides by 2048 samples at 96kHz (expect ≤ 1024)

Resolution: This bug fixed as of Pro Tools 11.0.1

12.56.3.134 PTSW-183902 / PT-218479

[AAX_IHostProcessorDelegate::GetAudio\(\)](#) responds to invalid iLocation as if everything succeeded

Resolution: This bug will not be fixed

12.56.3.135 PTSW-183848 / PT-218390

[AAX_IHostProcessorDelegate::GetAudio\(\)](#) ignores input audio buffer parameter

Resolution: This bug will not be fixed

The workaround for this issue is to make sure that HostProcessor plug-ins only request valid audio - do the boundary-condition checking inside the plug-in.

12.56.3.136 PTSW-183841

Plug-ins defining [AAX_eProperty_RequestsAllTrackData](#) quit when processing a timeline region with no audio

Resolution: This bug is fixed as of Pro Tools 11.0.2

12.56.3.137 PTSW-183731

Failures returned by 3P AAX-AS Pls in Pre- Analyze/Render are not used by the host

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.138 PTSW-183708

AudioSuite: plug-in parameters are not changed upon 1st click after you click Bypass. [Win]

Resolution: This was found to be an issue in certain plug-ins with JUCE-based GUI implementations. In JUCE, the real-time variants of the of the modifiers key getter method can cause seemingly unrelated problems with the responsiveness of the GUI. In this instance, the symptom was that plug-in parameters would not be changed on the first click inside the GUI window.

The workaround for this issue, and for other unusual GUI behavior in these plug-ins, is to always use `juce::ModifierKeys::getCurrentModifiers()`; do not use `juce::ModifierKeys::getCurrentModifiersRealtime()`.

12.56.3.139 PTSW-168222

Sample rate specific plug-ins cause Pro Tools to throw a misleading error message when opened in non-supported sample rate sessions

Resolution: This bug is fixed as of Pro Tools 11.1

12.56.3.140 PTSW-165992

Make automation link by Parameter ID instead of Parameter Name. Fall-back to Parameter Name if no match

Resolution: This behavior is supported starting in Pro Tools 11.1

12.56.3.141 PTSW-163739

AudioSuite works incorrectly in Clip List mode.

Resolution: This bug is fixed as of Pro Tools 12

12.56.3.142 PTSW-161674

Stereo instrument plug-ins: "MIDI Node" field in plug-in window header disappears when insert is dragged to a new slot

Resolution: This bug is unresolved in Pro Tools and will not be fixed for the foreseeable future.

12.56.3.143 PTSW-160778

After making a Preview pass, AudioSuite plug-ins no longer make calls to InitOutputBounds()

Resolution: This bug is fixed as of Pro Tools 10.2.1

12.56.3.144 PTSW-160620

AAX plug-ins receive meaningless Clock data on Native decks, and less-than-ideal data on DSP decks

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.145 PTSW-159702

AAX VI Issue - All AAX VIs do not have MIDI Nodes

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.146 PTSW-159700

AAX VI Issue - Instrument Tracks do not automatically map to the AAX VI that is instantiated on them

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.147 PTSW-159524

Incorrect error message when power is not connected to HDX card (EDIT: occurs with pre-A1 HDX prototypes only)

Resolution: This bug will not be fixed

12.56.3.148 PTSW-158119

Some plug-ins' DSP Instance counts are much lower in Pro Tools 10.2 than in Pro Tools 10.1

Resolution: This issue affects plug-ins that employ more than one buffered data port and that support many instances per DSP chip on HDX. As of Pro Tools 10.2, there is a limit of 164 buffered data ports per DSP (this is equal to the total I/O limit per DSP.)

To work around this issue, use as few data ports in your plug-in's algorithm context as possible. Note that DMA transfers on HDX occur in 128-byte chunks, so packet sizes below 128 bytes do not increase transfer efficiency on HDX.

12.56.3.149 PTSW-157745

Plug-ins write automation with pairs of updates, causing undesired "stepping" in recorded automation

Resolution: This bug is fixed as of Pro Tools 10.2 and 10.1.1

12.56.3.150 PTSW-157518

Poor plug-in performance with multiple processors selected; plug-ins are not consistently assigned to the same worker/thread by DAE, leading to cache thrashing.

Resolution: This bug is fixed as of the audio engine changes in Pro Tools 11

12.56.3.151 PTSW-157012

AAX DSP plug-ins with same DLL name are not properly labeled in the System Usage window

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.152 PTSW-156310

Mouse cursor does not reliably update when positioned over plug-ins. Instead the mouse cursor shows the current Edit Tool.

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.153 PTSW-156286

GUI elements fill window in some 3P AAX plug-ins GUIs on Windows

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.154 PTSW-156216

`pluginGestalt_SupportsControlChangesInThread` is not properly implemented for AAX plug-ins

Resolution: Parameter updates are handled by a non-main thread for all AAX plug-ins as of Pro Tools 10.1

12.56.3.155 PTSW-156195

Silent failure when plug-ins attempt to register components with different platform support

Resolution: This bug is not yet resolved. This is an expected constraint, but the silent failure is unexpected

12.56.3.156 PTSW-156035

`GetCurrentTDMSampleLocation()` returns the wrong value.

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.157 PTSW-155300 / PT-218458

When an AudioSuite plug-in modifies the output audio length, the audio is not positioned at the correct location

This bug is due to AudioSuite handles processing. A plug-in that modifies the output audio length may move audio from the handle region into the visible clip region, which is unexpected behavior from the user's perspective.

Resolution: This bug will not be fixed

The workaround is for plug-ins that experience this issue to disable AudioSuite handles, thereby only processing the audio that the user sees on the timeline.

12.56.3.158 PTSW-155177

`eFicGestalt_GetASPreHandleLength` and `eFicGestalt_GetASPostHandleLength` return the wrong handle length values upon a call to `AnalyzeAudio` with 'WHOLE FILE' mode selected

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.159 PTSW-154361

Highlight info sent to plug-ins before GUI is created.

Resolution: This bug is fixed as of Pro Tools 10.0

12.56.3.160 PTSW-153140

Crash on Pro Tools quit when plug-in GUI is open (macOS)

Resolution: This bug is unresolved in Pro Tools and will not be fixed for the foreseeable future. Plug-in workarounds are demonstrated in the `DemoGain_GUIExtensions` example plug-ins:

a) Separating all Obj-C elements into a separate bundle that is loaded manually by the main plug-in bundle (see `DemoGain_Cocoa`) b) Applying an `NSAutoreleasePool` to the AAX GUI object destructors (see `DemoGain_VST` and `DemoGain_JUCE`)

12.56.3.161 PTSW-150047

AAX MIDI plug-ins do not get correct MIDI routing on Instrument tracks

Resolution: This bug is fixed as of Pro Tools 10.2

12.56.3.162 PTSW-149880

Configurations with duplicate `PlugInID` properties are silently hidden with no error

Resolution: As of Pro Tools 10.2, duplicate `PlugInID` properties will trigger the following DigiTrace log:

`DTF_AAXHOST DTP_NORMAL`

"AAXH ERROR: Attempted to add new configuration with duplicate ID: %x" existingID

12.56.3.163 PTSW-149819

MIDI packet alignment is not identical between DAE and AAX

This is a known bug in Pro Tools 10.0. This bug results in corrupted MIDI stream data to AAX plug-ins.

Resolution: This bug is fixed as of Pro Tools 10.0.1

12.56.3.164 PTSW-135536 / PT-218412

Erroneous transport location information provided to plug-ins after playback (new to PT9)

Resolution: This bug will not be fixed

12.56.3.165 PTSW-3020 / PT-218463

Groups do not follow changes to "Inserts" Globals group settings

Resolution: This bug will not be fixed

The workaround for this issue is to modify the group's settings to de-select "follow globals", then re-modify the group's settings to select "follow globals". This will apply the current Globals settings as well as any future changes to the Globals without need for additional workarounds.

12.56.3.166 AAX-686

Re-add support for AudioSuite "progress" dialog re-naming (was supported in PT 9 and earlier) (see PTSW-159768)

Resolution: This bug is unresolved

12.56.3.167 AAX-583 / PTSW-157743

AAX SDK Win32 GUI example plug-in does not draw correctly

Resolution: Duplicate of PTSW-156286 (see above.) Resolved as of Pro Tools 10.2

12.56.4 Known Issues in Venue Live Sound Systems**12.56.4.1 VSW-13857**

Plug-in installers cannot associate a single thumbnail image with multiple variants

Resolution: This capability is supported starting in Venue 6.3

Prior to this change a plug-in thumbnail filename always contained 24 hexadecimal digits:

1. The first 8 digits refer to the plug-in's Manufacturer ID
2. The middle 8 digits refer to the plug-in's Product ID
3. The last 8 digits refer to the plug-in's "plug-in ID", which is usually a plug-in variant (Mono, Stereo, etc.).

With this change, Venue supports using a generic thumbnail file for all variants, thus having only the first 16 identifying digits.

12.56.4.2 VSW-13292

Plug-in parameters are not mapped to S6L if custom page tables are not provided

Details: S6L does not fall back to using the 'PgTL' page table type if a plug-in does not provide any parameter mapping for the primary 'Av46' or secondary 'FrTL' page tables. If a plug-in does not provide any of these page tables then its parameters will not display on the console.

Resolution: This bug will not be fixed

12.56.4.3 Other Known Issues

- [PT-210904 / VSW-14216](#)

12.56.5 Known Issues in Media Composer

12.56.5.1 MCDEV-2904

Optional analysis is not applied to every channel in a multi-channel selection

Resolution: This bug is fixed as of Media Composer 8.4

Discussion: When an optional analysis pass is triggered in Media Composer, only the channel that is currently represented in the AudioSuite Dialog will be analyzed. Other channels in a multi-channel selection will not be analyzed.

This issue is fixed in Media Composer 8.4; now the following behavior will occur for AudioSuite plug-ins:

- If a plug-in defines only [AAX_eProperty_RequiresAnalysis](#) then an Analyze pass will be performed before Render/Preview and the "Analyze" button will be disabled
- If a plug-in defines only [AAX_eProperty_OptionalAnalysis](#) then an Analyze pass will be performed before Render/Preview as well as when the "Analyze" button is clicked, and the "Analyze" button will be enabled
- If a plug-in defines both [AAX_eProperty_RequiresAnalysis](#) and [AAX_eProperty_OptionalAnalysis](#) then an Analyze pass will be performed before Render/Preview as well as when the "Analyze" button is clicked, and the "Analyze" button will be enabled

12.56.6 Known Issues in Control Surfaces

12.56.6.1 PT-285383

EUCON surfaces display plug-in parameters incorrectly if the plug-in page table contains an Av18 layout

Resolution: This bug is fixed in 2023.3

12.56.6.2 PT-226228

On EUCON control surfaces, Dynamics curves are not displayed if a plug-in does not provide a custom curve display range

Workaround: In order for a plug-in's Dynamics curve to be displayed, the plug-in must implement [AAX_IEffectParameters::GetCurveDataDisplayRange\(\)](#) for whichever Dynamics [curve types](#) it supports

12.56.6.3 PT-226227

EUCON control surfaces do not support custom EQ curve display ranges

Resolution: This feature is not yet implemented

12.56.6.4 GWSW-16656

Avid Control does not map plug-in controls to cells that do not have an encoder assignment

Resolution: This bug is unresolved

12.56.6.5 GWSW-8470

S6: Knob velocity changes are too sensitive for plug-in parameters with a large number of steps

Resolution: Resolved as of S6 Software 2.0

12.56.6.6 GWSW-6694

S6: Plug-in parameter order is inverted when using ProControl page tables

Resolution: Resolved as of S6 Software 1.3

12.56.7 Known Issues in Other Software**12.56.7.1 XPACE-23**

Performance issues on Azure VMs with some copy protected binaries

Pro Tools and Media Composer support operation in an Azure VM environment. Some early versions of Eden copy protection by PACE Anti-Piracy, Inc. does not perform well in this environment.

Resolution: This issue is resolved in PACE Eden versions 5 and later. To avoid this issue be sure to update to the latest version of your copy protection.

12.56.8 Known Issues in AAX Tools

For a list of known issues in AAX Tools such as Pro Tools Developer Builds, [DigiShell](#) or the [AAX Plug-In Page Table Editor](#), see the dedicated ReadMe file that is distributed with each tool. Collaboration diagram for Known Issues:



12.57 Change Log

Changes between AAX SDK versions.

12.57.1 Change Log

12.57.1.1 AAX SDK 2.6.1

12.57.1.1.1 Build

- Reduced default warning Visual Studio level to `Level4` to eliminate warnings from standard library headers

12.57.1.1.2 Definitions

- Added [AAX_eProperty_ShowInMenus](#) for plugins that should not be shown in the host's effect menus
- Added new plugin category [AAX_EPluginCategory_MIDIEffect](#) for MIDI Effect plugins

12.57.1.1.3 Example plug-ins

- The [DemoMIDI_Synth_AuxOutput](#) plugin now demonstrates stereo auxiliary outputs

12.57.1.1.4 Extensions

- Updated the JUCE GUI extension to support JUCE version 7

12.57.1.2 AAX SDK 2.6.0

12.57.1.2.1 AAX Library

- Added a implementation [AAX_CSessionDocumentClient](#) for the session document client interface
- Changed the behavior of [AAX_AggregateResult](#) to avoid crash on an exception thrown from the class destructor
- Added additional reference implementations of [AAX_IDataBuffer](#) - [AAX_CArrayDataBuffer](#) and [AAX_CArrayDataBufferOfType](#)

12.57.1.2.2 Definitions

- Added a new tracing priority, [kAAX_Trace_Priority_Critical](#)

12.57.1.2.3 Documentation

- Re-named TI Guide to [HDX DSP Guide](#)
- Clarified documentation for HDX DSP [Multi-shell packing](#)
- Added documentation regarding [Using the Pro Tools Scripting SDK with AAX](#)
- Added BNDL as a valid bundle ID in [AAX Format Specification](#)

12.57.1.2.4 Interface

- Added [AAX_IACFTransportControl](#) and [AAX_IACFTransport_V4](#), with methods typically accessed through [AAX_ITransport](#)
- Added [AAX_IACFSessionDocument](#), with methods typically accessed through [AAX_ISessionDocument](#), and [AAX_IACFSessionDocumentClient](#), with implementation usually performed via [AAX_CSessionDocumentClient](#)
- Added [AAX_IACFDataBuffer](#), with methods typically accessed through [AAX_IDataBuffer](#) when the data buffer is implemented on the host

See [Host Support](#) for host support information

12.57.1.2.5 Resolved bugs

- Added return value checks for internal calls to [AAX_IEffectParameters::GetNumberOfParameters](#) and [AAX_IEffectParameters::GetParameterIDFromIndex](#)

12.57.1.3 AAX SDK 2.5.1

12.57.1.3.1 AAX Library

- Updated [AAX_CParameter](#) so that the host is now automatically notified whenever [AAX_CParameter::SetName\(\)](#) is used.

12.57.1.3.2 Definitions

- Added [kAAX_ParameterIdentifierMaxSize](#) to define the maximum size of a [AAX_CParamID](#) c-string.

12.57.1.3.3 Interface

- Added [Task agent interface](#) including the [AAX_ITaskAgent](#) module and the generic [AAX_IDataBuffer](#) reference counted data container.
- Added [AAX_eNotificationEvent_ParameterNameChanged](#) notification type. Users of [AAX_CParameter](#) do not need to post this notification directly.

12.57.1.3.4 Utilities

- Replaced `snprintf` and fixed size buffers with C++ idioms in the [AAX_StringUtilities.h](#) utility functions

12.57.1.4 AAX SDK 2.5.0

12.57.1.4.1 Definitions

- Added [AAX_EStemFormat](#) definitions for surround stem formats from 5.0.2 to 9.1.6 and for Ambisonics formats from fourth to seventh order

12.57.1.4.2 Example plug-ins

- DemoGain_UpMixer has been updated with all combinations of the new stem formats

12.57.1.4.3 Interface

- New interfaces:
 - [AAX_IACFViewContainer_V3](#), with methods to track mouse movement over controls, accessed through [AAX_IViewContainer](#)

See [Host Support](#) for host support information

12.57.1.5 AAX SDK 2.4.1

12.57.1.5.1 Build

- Treat Warnings As Errors is now disabled for the [AAX](#) Library Xcode project
- The [AAX](#) Library Xcode project is no longer configured to use the Legacy Build System, which is deprecated in current Xcode

12.57.1.6 AAX SDK 2.4.0

12.57.1.6.1 Build

- Compilation for arm64 is now supported
- Explicitly set macOS project architectures to `x86_64` and `arm64`
- Updated Visual Studio project format to VS2017 and resolved newly detected warnings
- Reduced Visual Studio warning level from `EnableAllWarnigs` to `Level4` for the AAXLibrary project

12.57.1.6.2 Definitions

- Added [AAX_eProperty_AlwaysBypass](#) for plug-ins that always pass the audio signal through unaltered
- Added [AAX_eProperty_ObservesTransportState](#) , [AAX_eNotificationEvent_TransportStateChanged](#) , [AAX_ETransportState](#) , [AAX_ERecordMode](#) , and [AAX_TransportStateInfo_V1](#) to provide information about the current state of the host transport
- Added [AAX_eNotificationEvent_LogState](#) as an optional logging convenience mechanism for certain plug-ins that use the Direct Data feature
- Extended [AAX_EFrameRate](#) to include additional frame rates. These additional rates can be queried using [AAX_ITransport::GetHDTIMECodeInfo\(\)](#)
- Added [AAX_ERROR_PRINT_FAILURE](#) for printing library method failures

12.57.1.6.3 Documentation

- Added the [Real-time performance](#) page and the [Plug-In Causes Audio Streaming Errors](#) troubleshooting section with overview of best practices for avoiding streaming errors and achieving good performance for audio processing on real-time threads
- Updated the [HDX DSP Guide](#) page with a "Getting Started" section and with information about Pro Tools | Carbon
- Corrected the documented range of acceptable values between [AAX_ERROR_PLUGIN_BEGIN](#) and [AAX_ERROR_PLUGIN_END](#)
- Improved Doxygen dot image resolution, now using SVG

12.57.1.6.4 Example plug-ins

- `DemoGain_UpMixer` now registers non-converting combinations

12.57.1.6.5 Interface

- New interfaces:
 - [AAX_IACFEfffectDirectData_V2](#), with methods accessed through [AAX_IEffectDirectData](#)
 - [AAX_IACFTransport_V3](#), with methods typically accessed through [AAX_ITransport](#)

See [Host Support](#) for host support information

12.57.1.6.6 Resolved bugs

- Fixed [AAXSDK-705](#)

12.57.1.6.7 Utilities

- CreatePackage.bat now removes the read-only attribute from the .aaxplugin folder on Windows

12.57.1.7 AAX SDK 2.3.2

12.57.1.7.1 AAX Library

- Removed unnecessary `virtual` keyword usage for method overrides
- Removed unused `mClipped` member of [AAX_CEffectParameters](#)
- Convert `mViewContainer` member of [AAX_CEffectGUI](#) to a smart pointer

12.57.1.7.2 Build

- Xcode 10 and Visual Studio 2017 are now supported
- Added Xcode workspace and Visual Studio solution containing all projects in the AAX SDK for convenience
- Removed 32-bit architecture targets from all project configurations. 32-bit architectures are still supported by AAX if you choose to explicitly add them to your build project configurations.
- Updated all Xcode projects to recommended `CFBundleIdentifier` usage and build settings

12.57.1.7.3 Definitions

- Added [AAX_eProperty_Constraint_DoNotApplyDefaultSettings](#) for plug-ins which need to disable the normal default settings application procedure used by Pro Tools
- Added [AAX_eNotificationEvent_PriorSettingsInvalid](#) which may be useful for certain plug-ins when running in Venue systems
- Added [AAX_eProperty_PluginID_NoProcessing](#) for Effect types that do not process audio
- Removed [AAX_eProperty_SupportsProgressDialog](#) which is not supported in any AAX host

12.57.1.7.4 Documentation

- Fixes and improvements in the "Plug-In spec properties" section of [AAX_Properties.h](#)
- Added [Quick Start](#) and [Troubleshooting](#) documentation sections
- Added additional detail to the [Digital signature](#) section of the [Pro Tools Guide](#)

12.57.1.7.5 Example plug-ins

- Changed ID generation algorithm for [DemoGain_UpMixer](#). Older copies of this example plug-in will not be recovered in saved sessions.

12.57.1.7.6 Resolved bugs

- Fixed [AAXSDK-663](#) AAX SDK `#pragma pack` errors with XCode 10 and later

12.57.1.8 AAX SDK 2.3.1

12.57.1.8.1 AAX Library

- Enhanced support for [AAX_CheckedResult](#) - added [AAX_CAPTURE](#), [AAX_CAPTURE_MULT](#), and [AAX_AggregateResult](#) to assist with common Describe error handling scenarios
- Updated [AAX_CMonolithicParameters::StaticDescribe\(\)](#) to use [AAX_CheckedResult](#) for error checking
- Added [AAX_CStatelessParameter](#) for "momentary" parameters which do not require state, such as tap tempo buttons which can be mapped to a control surface
- Improved tolerance for unknown parameters when building or parsing plug-in settings chunk data
- Added [AAX_DEBUGASSERT](#), [AAX_STACKTRACE](#), and [AAX_TRACEORSTACKTRACE](#) to the library of tracing and assertion macros in [AAX_Assert.h](#)
- Fixed warnings which would prevent compilation in Visual Studio 2015 and Visual Studio 2017 when Treat Warnings As Errors is enabled
- Removed Visual Studio 2008, Visual Studio 2010, and Xcode 3 projects

12.57.1.8.2 Definitions

- Removed guard preventing [AAX_CPP11_SUPPORT](#) from being set for PACE Fusion compiler builds
- Deprecated [AAX_EHostMode](#) - replaced by [AAX_EHostModeBits](#)

12.57.1.8.3 Documentation

- Documentation added to [EQ and Dynamics Curve Displays](#) for the EQ Curves feature in Pro Tools 2018.1
- Added [Checking Results](#) and [Describe Validation](#) sections to [Description callback](#)
- Added [Building your plug-in installer](#) section to [Distributing Your AAX Plug-In](#), including information about bundling Track Presets with the plug-in installer

12.57.1.8.4 Example plug-ins

- Updated all example plug-ins' Describe routines to use [AAX_CheckedResult](#) for error checking
- Updated some example plug-ins' parameter registration code in [EffectInit\(\)](#) with a safer parameter creation and release style using `std::unique_ptr`
- Updated the [RectiFi](#) example plug-in to match the current shipping version of Avid's Recti-Fi plug-in
- [DemoGain_UpMixer](#) now converts arbitrarily between all stem formats, both wider and narrower
- Removed Visual Studio 2008, Visual Studio 2010, and Xcode 3 projects
- Common Xcode settings updated with "macosx10.11" base SDK and "10.9" deployment target
- Added [AAX](#) DSP for higher stem formats in [DemoGain_Multichannel](#) and [DemoGain_UpMixer](#)

12.57.1.8.5 Extensions

- Updated the VSTGUI extension and example plug-in to use VSTGUI 4.3

12.57.1.8.6 Interface

- New interfaces:
 - [AAX_IACFPageTable_V2](#), with methods accessed through [AAX_IPageTable](#)
 - [AAX_IACFHostServices_V3](#), with methods typically accessed through the macros in [AAX_Assert.h](#)

See [Host Support](#) for host support information

12.57.1.8.7 Utilities

- Added reference count tracing logic to [AAX_CACFUnknown.cpp](#), which can be toggled on using the `AAX_↔DEBUG_ACF_REFCOUNT` macro
- Added some convenience functions to [AAX_PageTableUtilities.h](#)
- Added [getLowestSampleRateInMask\(\)](#) and [getMaskForSampleRate\(\)](#) convenience functions

12.57.1.9 AAX SDK 2.3.0

12.57.1.9.1 AAX Library

- Added [AAX_Exception.h](#) with the [AAX::Exception](#) namespace for AAX-specific exception objects and the [AAX_CheckedResult](#) class which can be used for throwing AAX exceptions when an error is encountered.
- Added a try/catch block in the library implementation of [AAXRegisterPlugin](#) such that exceptions may safely be thrown during Describe
- [AAX_ICollection](#) now provides convenience methods to access an [AAX_IDescriptionHost](#) and [IACFDefinition](#), if these interfaces are supported by the host during Describe
- [AAX_IComponentDescriptor](#) now provides the generic [AddProcessProc\(\)](#) method for specifying multiple ProcessProcs at once using a property map
- [AAX_IController](#) now provides methods for copying page table data from other effect variants or from arbitrary page table files on disk
- [AAX_IPropertyMap](#) now supports pointer-sized properties
- [AAX_IPropertyMap](#) objects can now be generated from other property map objects without requiring access to a component factory interface

12.57.1.9.2 Definitions

- Added a new stem format definition for the [7.0.2](#) format
- Removed the previous FuMa Ambisonics formats and added definitions for [second-order](#), and [third-order](#) ACN Ambisonics stems
- Added new notification types:
 - [AAX_eNotificationEvent_ParameterMappingChanged](#) (plug-in to host)
 - [AAX_eNotificationEvent_HostModeChanged](#) (host to plug-in)
- C++11 keyword compatibility macros added to [AAX.h](#)
- Removed the `AAX_AlignedDouble` definition, which was unused

12.57.1.9.3 Documentation

- New documentation:
 - [Distributing Your AAX Plug-In](#)
 - [EQ and Dynamics Curve Displays](#)
 - [Adding signposts to the DigiTrace log at run-time](#)
 - [Plug-in preset data comparison](#) for Media Composer
 - [Interactive mode](#) for DTT
 - Descriptions of [Pro Tools | Control app and Pro Tools | Dock](#) in the [Page Table Guide](#)
- There is a new process for [requesting the digital signing toolkit](#) for digitally signing AAX plug-ins
- Added a PDF print-out of this Doxygen documentation to assist with text-based searches
- Updated the [Contacting Avid](#) section of the main page to clarify the various processes for communicating with Avid
- Updated [AAX_Errors.h](#) with a list of current internal AAX host error values, which are useful for reference when troubleshooting host errors.
- Updated the [HDX DSP Guide](#) with information about using the latest version of Code Composer Studio with this AAX SDK

12.57.1.9.4 Example plug-ins

- Base Mac OS SDK setting in the common .xcconfig files is now macosx10.9
- DemoGain_Multichannel now includes an example of gain reduction metering
- DemoGain_Multichannel now supports [7.0.2](#) and [First-order](#), [second-order](#), and [third-order](#) Ambisonics stem formats
- DemoGain_UpMixer example plug-in added to demonstrate a width-changing effect
- The DemoMIDI_NoteOn example plug-in algorithm now supports note hold

12.57.1.9.5 Interface

- New interfaces:
 - [AAX_IACFComponentDescriptor_V3](#), with methods accessed through [AAX_IComponentDescriptor](#)
 - [AAX_IACFDescriptionHost](#), with methods accessed through [AAX_IDescriptionHost](#)
 - [AAX_IACFEfffectParameters_V4](#), with methods accessed through [AAX_IEffectParameters](#)
 - [AAX_IACFFeatureInfo](#), with methods accessed through [AAX_IFeatureInfo](#)
 - [AAX_IACFPageTable](#), with methods accessed through [AAX_IPageTable](#)
 - [AAX_IACFPageTableController](#), with methods accessed through [AAX_IController](#)
 - [AAX_IACFPropertyMap_V3](#), with methods accessed through [AAX_IPropertyMap](#)

See [Host Support](#) for host support information

- Added the concept of a host "feature" which can be queried during Describe execution using [AAX_IDescriptionHost](#) and [AAX_IFeatureInfo](#)

12.57.1.9.6 Resolved bugs

- Resolved [AAXSDK-533](#): AAXLibrary compiles with warnings in VS2015 / VS2017
- Resolved [AAXSDK-514](#): Using collection-level properties leads to a leaked ACF object
- Fixed bugs with taper delegates when the minimum and maximum values are equal
- Some unnecessary headers removed or converted to forward declarations

12.57.1.10 AAX SDK 2.2.2

12.57.1.10.1 AAX Library

- Added new methods to [AAX_IParameter](#) for easier conversion between logical and normalized parameter values
- Re-named `AAX_CParameterManager::ControlIndexFromID()` to [AAX_CParameterManager::GetParameterIndex](#)
- Added AAX Library project for Visual Studio 2013
- Added warning exclusion for C4738 to 32-bit Release configuration of the AAX Library project on Windows to fix a treat-warnings-as-errors build failure that can occur in this configuration when linking statically to the MSVC run-time libraries

12.57.1.10.2 Definitions

- Added new stem format selectors for the following stem formats:
 - The [7.1.2](#) speaker configuration
 - [First-order](#), [second-order](#), and [third-order](#) Ambisonics
- Added a new notification type for information regarding the host's delay compensation state↔: [AAX_eNotificationEvent_DelayCompensationState](#)
- Added a new [input data port type](#) property for ports which request [incrementally-buffered](#) packet delivery
- Added a property to allow different AAX DSP plug-in types to share the same DSP chip even if [AAX_eProperty_TI_MaxInstancesPerChip](#) is declared: [AAX_eProperty_TI_ForceAllowChipSharing](#)

12.57.1.10.3 Documentation

- Added specific details about display hardware to the [VENUE Guide](#)

12.57.1.10.4 Example plug-ins

- Added the [DemoGain_Multichannel](#) example plug-in
- Updated page tables of all example plug-ins
- Example plug-in Xcode projects now use C++11 and libc++ by default
- Updated [DemoDelay_Hybrid](#) to fix problems with instantiation in [DSH](#) and other test hosts
- Removed multi-mono support from [DemoMIDI_Synth](#) to provide a better example of a standard VI configuration
- Updated [Recti-Fi](#) example plug-in IDs so that they will not collide with the shipping version of Recti-Fi

12.57.1.10.5 Extensions

- Updated `AAX_JuceContentView::mouseMove()` for compatibility with Juce version 4 and higher
- Updated `AAX_CEffectGUI_VST` for compatibility with 32-bit plug-ins when used with VSTGUI 4.2

12.57.1.10.6 Interface

- ACF interface files updated to a more recent version of the ACF SDK

12.57.1.10.7 Resolved bugs

- [AAX_CEffectParameters::UpdateParameterNormalizedValue\(\)](#) now increments the effect change counter only when the parameter's value actually changes

12.57.1.10.8 Utilities

- New utility functions: [AAX::AsStringStemFormat\(\)](#), [AAX::AsStringStemChannel\(\)](#)
- Added [AAX_SCOPE_COMPUTE_DENORMALS\(\)](#) for forcing denormal float values to be calculated within a scope, rather than being treated as zero (currently implemented for Mac only)

12.57.1.11 AAX SDK 2.2.1

12.57.1.11.1 Interface

- New interfaces:
 - [AAX_IACFController_V3](#)

12.57.1.11.2 Documentation

- Added the [VENUE Guide](#) page
- Updated the [Page Table Guide](#)
 - Updated VENUE information: Added information about [VENUE | S6L](#) and [VENUE | S3L-X](#) and removed information about VENUE systems which do not support AAX plug-ins
 - Added information for S6, including details about the 'Av46' page table type and a new section on [Center Section Parameter Mapping in S6 Expand Mode](#)
- Updated the documentation for [Plug-in type conversion](#), including a new section describing [Type deprecation](#)
- Fixed image display problems on the [DSH Guide](#) page
- Added pre-built HDX DLL files to the SDK for all example plug-ins which support AAX DSP

Note

The example plug-ins' Visual Studio projects now include a `PostBuildEvent` command which will copy the plug-in's HDX DLL from the project's `TI/bin/Release` folder to the built .aaxplugin's `Resources` folder.

- Additional minor example plug-in fixes
 - Removed unnecessary build phases and framework dependencies from the plug-ins' Xcode projects
 - Removed "%AAX" from the example plug-ins' display names
 - Changed the guard for AAX DSP cycle count declarations to check for the definition of the `AAX_↔ TI_BINARY_IN_DEVELOPMENT` preprocessor symbol before adding cycle counts to the plug-in's description
 - Added "example" to the names of all example plug-ins

12.57.1.11.3 AAX Library

- Extended [AAX_CParameter::GetValueAsString\(\)](#) and [AAX_CParameter::SetValueWithString\(\)](#) with support for all value types
- Fixed the specialization of [AAX_CPacket::GetPtr\(\)](#) for `void*` so that it is called when the `void*` version of the function template is requested

12.57.1.11.4 Definitions

- Added [AAX_ePlugInStrings_ClipNameSuffix](#)
- Added a definition of the `TI_VERSION` preprocessor macro for the TI DSP compiler in [AAX.h](#)

12.57.1.12 AAX SDK 2.2.0

12.57.1.12.1 Interface

- New interfaces:
 - [AAX_IACFEfffectParameters_V3](#)
 - [AAX_IACFHostProcessor_V2](#)
 - [AAX_IACFHostProcessorDelegate_V3](#)
 - [AAX_IACFHostServices_V2](#)
 - [AAX_IACFViewContainer_V2](#)

12.57.1.12.2 Directory changes

- Moved common processing classes for the SDK example plug-ins to ExamplePlugIns/Common/Processing↔Classes
- Moved MIDI logging utilities to the Extensions folder
- Moved [AAX_CMonolithicParameters](#) to the Extensions folder and removed it from the AAX Library

12.57.1.12.3 Extensions

- Changed VST project to use the newest version of VSTGUI sources - VSTGUI 4.2
- Created Visual Studio 2012 projects for GUI Extensions
- Fixed [AAX_CMonolithicParameters](#) so that it correctly supports [AAX_eConstraintLocationMask_DataModel](#)

Note

This value is **required** for all plug-ins that share memory between their data model and algorithm call-back

- Updated [AAX_CMonolithicParameters](#) to include parameter value synchronization
- Updated [AAX_CMonolithicParameters](#) to support Hybrid and include a state counter field

12.57.1.12.4 Definitions

- Changed name of `AAX_eProperty_StoreXMLPageTablesByType` to [AAX_eProperty_StoreXMLPageTablesByEffect](#) to best reflect the actual behavior of this property
- Replaced [AAX_EPlugInCategory_Effect](#) category (erroneously removed in AAX SDK 2.1)

12.57.1.12.5 Utilities

- Added utilities for atomic operations and a thread-safe FIFO queue class: [AAX_CAtomicQueue](#)
- Added AAX stacktrace logging support to make plug-in debugging easier: see [AAX_STACKTRACE](#) and [AAX_TRACEORSTACKTRACE](#)
- Added a utility for locating the .aaxplugin bundle to provide an ability to access resources in the bundle

12.57.1.12.6 AAX Library

- Created an AAX Library project for Visual Studio 2012
- Created a libc++ target in the AAX Library Xcode project
- Resolved "incompatible ms_struct" warning in Xcode 6; removed [AAX_ALIGN_FILE_ALG](#) from inappropriate locations such as virtual classes that do not cross library boundaries
- Added [AAX_IParameterValue](#), an abstract value class for parameter data, and refactored [AAX_CParameter](#) to use this interface
- Re-named `AAX_CInstrumentParameters` to [AAX_CMonolithicParameters](#) (see the [Extensions](#) section for more information)
- Added an [AAX_CStateDisplayDelegate](#) constructor taking `std::vector<AAX_IString*>`
- Added an [AAX_CParameter](#) constructor taking [AAX_IString](#) as an identifier
- Added hex conversion methods to [AAX_CString](#)
- Fixed chunk size error handling in [AAX_CChunkDataParser](#)

12.57.1.12.7 Example plug-ins

- Added [DemoMIDI_Synth](#) and [DemoMIDI_Synth_AuxOutput](#) plug-ins
- Created Visual Studio 2012 projects for all example plug-ins
- Added EUCON page tables for all example plug-ins
- Various fixes for modifier-click event handling in example plug-ins
- Updated the example plug-in projects so that all built plug-in bundle filenames include "_Example"
- Corrected input/output property usage in HostProcessor example plug-ins
- Fixed multi-channel processing in [DemoDelay_HostProcessor](#)
- Fixed a bug with dynamic processing in [DemoMIDI_NoteOn](#) example plug-in
- Fixed [DemoGain_GUIExtensions](#) Win32 example plug-in GUI so that it is correctly displayed in Windows 8

12.57.1.12.8 Documentation

- Added [Media Composer Guide](#)
- Updated [Host Support](#) documentation for latest AAX host versions
- Updated the [Page Table Guide](#)
 - Updated [EUCON Page Tables](#) documentation
 - Updated [Avid Center Section Page Tables](#) documentation with tables mapping the EQ, Comp/Lim, and Exp/Gate table indices to their respective functions
- Updated [MIDI node](#) documentation
- Added new documentation pages for [Parameter update timing](#) and [Parameter automation](#)
- Improved [Presets and settings management](#) documentation
- Documented the [plug-in caching](#) behavior in Pro Tools
- Added documentation for optimizing an AAX DSP plug-in by using a hard-coded buffer size in the algorithm callback/ See the [Refactoring conditionals and branches](#) section of the [HDX DSP Guide](#)

12.57.1.13 AAX SDK 2.1.1

12.57.1.13.1 Definitions

- Explicitly removed support for the SDK's C99Compatibility headers in Microsoft Visual C++ 10.0 and later

12.57.1.13.2 DSP

- Added support and documentation for compiling AAX DSP plug-ins using Code Composer Studio 5
- Updated all example plug-in projects for use with Code Composer Studio 5

12.57.1.13.3 Documentation

- Extended the [parameter update documentation pages](#) with sequence diagrams and further information about linked parameter behavior
- Added guides for [DigiTrace](#) and [DSH](#)
- Added a reference list of [AAX interfaces](#)

12.57.1.14 AAX SDK 2.1.0

12.57.1.14.1 Interface

- New method added to [AAX_IACFTransport_V2](#) : [IsMetronomeEnabled\(\)](#)

12.57.1.14.2 AAX Library

- New methods in [AAX_CString](#) for direct copy from, assignment to, and comparison with `std::string`
- Fixed many implicit sign conversions
- Added `const` qualification to some [AAX_C...](#) methods
- Updated [AAX_IParameter::GetValueAsString\(\)](#) to take a pointer-to [AAX_IString](#) (was lvalue ref)
- Fixed a bug in [AAX_CEffectParameters::GetParameterNameOfLength\(\)](#); the method now correctly truncates a parameter name if the requested length is shorter than the shortest available abbreviated name
- Treat Warnings As Errors enabled in AAX Library projects
- clang pragmas added to avoid warnings for non-virtual destructors in ACF interface classes (cf. Microsoft COM)
- Xcode 3 project added for the AAX Library

12.57.1.14.3 Definitions

- Alignment of [AAX_CMidiPacket](#) and [AAX_CMidiStream](#) on 32-bit macOS is now explicitly set using `#pragma options align=power` to maintain backwards-compatibility with earlier versions of Pro Tools
- New property added: [AAX_eProperty_RequiresChunkCallsOnMainThread](#)
- New property added: [AAX_eProperty_Constraint_AlwaysProcess](#)
- Converted [AAX_eProperty_Related_Plugin_List](#) (property #22) to dedicated [DSP](#) and [Native](#) versions
- Re-named [AAX_eProperty_AudioBufferLength](#) to [AAX_eProperty_DSP_AudioBufferLength](#)
- Added new [AAX_ECurveType](#) selector: [AAX_eCurveType_Reduction](#)
- Added various new selectors to [AAX_ENotificationEvent](#)
- Updated [AAX_STEM_FORMAT](#) macros to allow negative index values
- Added new error codes to [AAX_EError](#)

12.57.1.14.4 Utilities

- New utility functions: [AAX::IsAvidNotification\(\)](#), [AAX_IsASCII\(\)](#), [AAX_AsStringFourChar\(\)](#)
- [AAX_ASSERT](#) and [AAX_TRACE](#) now require a trailing semicolon
- Re-named `LIMIT` to [AAX_LIMIT](#)
- Removed unused extended-80 conversion utilities

12.57.1.14.5 Extensions

- Resolved issue in which VSTGUI v4 key events were not received on Windows
- Xcode 3 projects added for the Juce and VSTGUI extension libraries

12.57.1.14.6 Documentation

- .pdf documentation moved to Doxygen
- Added several new sample plug-ins
- Expanded documentation for Host Processor and AAX Hybrid

12.57.1.15 AAX SDK 2.0.1

12.57.1.16 AAX SDK 2.0.0

12.57.1.16.1 AAX Library

- Added support for the AAX Hybrid processing architecture
- Added methods for better access to global MIDI data from [AAX_IEffectParameters](#)
- Extended the [AAX_ITransport](#) interface with several new methods
- Host Processor plug-ins can now trigger an analysis pass programmatically

12.57.1.16.2 Definitions

- Added new selectors to [AAX_ENotificationEvent](#) for state information during AudioSuite, bounce, and restore events
- AudioSuite reverb and delay plug-ins may opt out of the "Reverse" processing mode

12.57.1.16.3 Algorithm

- Support for temporary algorithm data blocks

12.57.1.17 AAX SDK 1.5.0**12.57.1.17.1 AAX Library**

- Plug-ins now receive a different notification when receiving chunks from session and preset loads
- Aux output stems now support up to 256 output channels
- Added alpha versions of V2 interfaces
- Added projects for Visual Studio 2005 and 2008

12.57.1.18 AAX SDK 1.0.6**12.57.1.18.1 Documentation**

- 64-bit targets enabled for the AAX Library and sample plug-ins

12.57.1.18.2 AAX Library

- Changed scope of some methods in [AAX_CEffectParameters](#) and [AAX_CEffectGUI](#)
- New 8 byte structure alignment added to [AAX.h](#)
- Changed the scope of some chunk parser items
- Clock context field is set to be synchronized across multiple plug-in instances
- Support for multiple input MIDI nodes
- Support for multiple named Aux Outputs ([AAX_CInstrumentParameters](#))
- Instrument parameters no longer uses host generated GUI by default

12.57.1.18.3 DSP

- Algorithm initialization routine now has 5 seconds to execute

12.57.1.19 AAX SDK 1.0.5**12.57.1.19.1 Directory Changes**

- Removed 3 files in /ExamplePlugIns/Common
- Added [AAX_UtilsNative.h](#) and [AAX_Version.h](#)
- Moved [AAXLog\(\)](#), [AAXLogf\(\)](#), and [isParameterIDEqual\(\)](#) to [AAX_UtilsNative.h](#)

12.57.1.19.2 Documentation

- Fixed instance tracking bugs in DemoGain_BackGround
- Added a time-stamp parameter to DemoMIDI_NoteOn
- Added MIDI-through to DemoMIDI_NoteOn
- Added DemoGain_DMA sample plug-in

12.57.1.19.3 AAX Library

- Changed scope of some methods in [AAX_CEffectParameters](#)
- Set default number of steps in [AAX_CParameter.h](#) to non-zero
- Renamed enum AAX_EConstraintLocation to AAX_EConstraintLocationMask

12.57.1.19.4 DSP

- Larger buffer size allowed on TI
- Support for DLL chip affinity in Pro Tools 10.2 and higher
- New AAX_INT_LO and AAX_INT_HI utilities defined

12.57.1.20 AAX SDK 1.0.4

12.57.1.20.1 Describe

- Multi-mono support constraint property added
 - Will be supported in DAE versions 10.2 and higher

12.57.1.20.2 AAX Library

- AAX_CInstrumentParameters class added as helper for monolithic instruments
- AAX_CTimestamp type changed to signed 64-bit integer
- Maximum string length support added to binary display delegate

12.57.1.20.3 Documentation

- Resolved several DemoGain_GUIExtensions example plug-in bugs and improved parity with expected Pro Tools plug-in GUI features
- Added DemoMIDI_Sampler example plug-in
- Added /TI/SignalProcessing directory with example signal processing utilities
- Added new "AAX for Pro Tools" document (still in progress)

12.57.1.21 AAX SDK 1.0.3

12.57.1.21.1 Describe

- Added "deprecated type" feature for swapping in new Effect types
- Removed AAX_eProperty_TI_UncachedCycleCount
- Removed AAX_eProperty_UseSmallPreviewBuffer, as this property is now mandatory

12.57.1.21.2 Algorithm

- Established 1024 as the maximum expected audio buffer length for any AAX plug-in
- Created new instance initialization action flag for instance reset events

12.57.1.21.3 AAX Library

- Fixed reference-counting bug in [AAXRegisterPlugin\(\)](#)

12.57.1.21.4 DSP

- Extra software pipeline information added to CCS asm output by default
- External memory support added to default CommonPlugIn_LinkerCmd.cmd file
 - ExtendedPlugIn_LinkerCmd.cmd is now deprecated

12.57.1.21.5 Utilities

- DigiTrace facility for AAX_Assert changed from DTF_TIPLUGINS to DTF_AAXPLUGINS
- Added example DTT script for signal cancellation testing to Development builds
- Added DSP information tooltip feature to plug-in window header in Pro Tools

12.57.1.21.6 Documentation

- Win32 GUI example plug-in added to the SDK
- Basic coefficient smoothing example plug-in added to SDK
- Side Chain and Auxiliary Output Stem information page added to Doxygen
- Resolved SetControlHighlightInfo() naming inconsistency in sample plug-ins
- Expanded GUI information in AAX Manual

12.57.1.22 AAX SDK 1.0.2

12.57.1.22.1 AAX Library

- Moved AAX Library source to /Libs directory
- Added complete library source code and project files
- Removed pre-compiled AAX library binaries

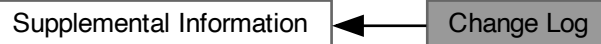
12.57.1.22.2 Documentation

- Added correct mouse event handling logic to DemoGain_GUIExtensions plug-ins
- Added meters to DemoGain_Cocoa
- New TI optimization case studies added to the TI Guide document

12.57.1.22.3 Resolved bugs

- PTSW-149745
 - Loading code into external DSP memory is functional as of TI Shell build 10.1x828

Collaboration diagram for Change Log:



12.58 Example Plug-Ins

Descriptions of the SDK's example plug-ins.

12.58.1 SDK Example plug-ins

This SDK includes the following example plug-ins. These plug-ins are designed to demonstrate good AAX plug-in design with varying levels of complexity.

In general, the SDK includes one basic version of each example plug-in, as well as multiple variations on this basic version. Each of these variations demonstrates a particular feature or design approach. To see the specific changes that were made to implement a feature, compare the example plug-in variant that demonstrates the feature to the basic version of the plug-in.

Aside from the GUI Extension examples, which are designed to work with third-party GUI frameworks, each sample plug-in should successfully compile "out of the box". However, you may receive compilation errors during the plug-ins' post-build copy step due to the fact that compiled TI DLLs are not included with this SDK.

12.58.1.1 Basic examples

These plug-ins provide complete working examples of AAX plug-ins without a lot of extra features. Use these plug-ins as a starting point for understanding [AAX](#).

12.58.1.1.1 DemoGain DemoGain is the simplest example plug-in, incorporating a mono algorithm with gain and bypass parameters.

12.58.1.1.2 DemoDist DemoDist demonstrates some more sophisticated techniques such as coefficient calculation and packaging, private data allocation, and multiple stem format support. DemoDist also demonstrates some basic optimization strategies for improving real-time algorithmic performance.

12.58.1.1.3 DemoDelay DemoDelay implements a basic delay algorithm. The variants of this example demonstrate a variety of alternative processing features provided by [AAX](#).

12.58.1.1.4 DemoMIDI_NoteOn DemoMIDI_Note on demonstrates basic MIDI input functionality. The example will create a step function with every Note On and Note Off message it receives. It also shows how to handle MIDI packages in the Data Model by overriding the [AAX_CEffectParameters::UpdateMIDINodes\(\)](#) method.

12.58.1.1.5 RectiFi This is a fully ported version of the Recti-Fi plug-in from Avid's D-Fi suite. For more information about Recti-Fi, see <http://www.avid.com/plugins/d-fi>

Note

The SDK's Recti-Fi example plug-in is currently out of date and does not accurately represent Avid's shipping Recti-Fi plug-in.

12.58.1.2 Feature examples

Each of these plug-ins is a slight variation on one of the [Basic examples](#). Each feature example plug-in demonstrates a specific feature or a possible alternative design approach for the plug-in. Compare these plug-ins with the corresponding basic example plug-in when you want to understand how a feature or design should be applied to your own AAX plug-ins.

12.58.1.2.1 DemoGain_GUIExtensions These examples demonstrate the use of various native and third-party GUI frameworks with [AAX](#). The examples that use third-party frameworks are configured to link to static libraries that combine the SDK's [GUI Extensions](#) (located in /Extensions/GUI) and the applicable third-party GUI framework. These libraries are not included in the SDK, and you will need to install the applicable framework SDK before it will be possible to compile these example plug-ins.

Note

See bug [AAXSDK-599](#)

12.58.1.2.2 DemoGain_LinkedParameters This example demonstrates parameter linking. The plug-in is a stereo version of DemoGain, with options to link the left and right channels in two different modes.

12.58.1.2.3 DemoGain_Smoothed This example demonstrates efficient algorithmic coefficient smoothing using a slight variation on the basic DemoGain plug-in algorithm.

12.58.1.2.4 DemoGain_Background This example demonstrates a background routine for algorithm processing. This example also uses the AAX [direct data interface](#) for communicating algorithmic delay to the plug-in's controller.

12.58.1.2.5 DemoGain_DMA This example includes two Effects that demonstrate use of the Scatter/Gather and Burst DMA facilities in [AAX](#).

12.58.1.2.6 DemoGain_Multichannel This example demonstrates a multichannel plug-in configuration supporting all available point source stem formats.

This plug-in also includes a simple example of gain-reduction metering, which can be used to test host features which use this data such as the [gain reduction meters](#) in Pro Tools.

12.58.1.2.7 DemoGain_UpMixer This example demonstrates conversion between different stem formats

12.58.1.2.8 DemoGain_ParamValueInfo This example demonstrates an implementation of the [GetParameterValueInfo\(\)](#) method, which is used to properly display certain parameter details on attached control surfaces. See [Avid Center Section Page Tables](#) in the [Page Table Guide](#).

12.58.1.2.9 DemoDist_GenCoef This example demonstrates an alternative approach to parameter update handling. It bypasses the packet dispatcher helper class and directly overrides [UpdateParameterNormalizedValue\(\)](#) and [GenerateCoefficients\(\)](#). This approach may be appropriate for plug-ins that involve complex mapping between parameter updates, coefficient generation algorithms, and coefficient data packets.

12.58.1.2.10 DemoDelay_HostProcessor This example includes two Effects that demonstrate the optional [Offline processing interface](#) for advanced offline processing features. One Effect implements a simple offline delay line, while the other Effect implements the same delay line but compensates for its delay when rendering to the timeline. This demonstrates how to manually compensate for inherent algorithmic delay in an offline processor.

Note

The output of offline plug-ins that do not use the [Offline processing interface](#) will be automatically adjusted by the host to account for any declared latency. The manual compensation technique demonstrated by DemoDelay_HostProcessor is **only** necessary in plug-ins that implement the [Offline processing interface](#), e.g. plug-ins that require nonlinear offline processing features.

12.58.1.2.11 DemoDelay_Hybrid This example demonstrates the optional [Hybrid Processing architecture](#) architecture for AAX plug-ins. This plug-in implements a short delay line that is rendered in the high-latency hybrid context. It can be built and run for either AAX Native or AAX DSP.

12.58.1.2.12 DemoDelay_DynamicLatencyComp This example demonstrates how to properly handle algorithmic latency changes at run-time. It uses a delay line to emulate a latency-inducing algorithm with varying latency based on the delay parameter setting. When the plug-in's latency compensation feature is enabled it declares this latency to the host.

Host Compatibility Notes The DemoDelay_DynamicLatencyComp example is compatible with Pro Tools 11.1 and higher.

12.58.1.2.13 DemoMIDI_Synth A basic synthesizer plug-in demonstrating use of an external object to manage the plug-in's state. AAX Native plug-ins that are designed to work with a cross-format framework may use a similar design. This plug-in uses [AAX_CMonolithicParameters](#) and therefore is AAX Native only.

12.58.1.2.14 DemoMIDI_Synth_AuxOutput A variation on [DemoMIDI_Synth](#) demonstrating the [Auxiliary Output Stems](#) feature. This instrument plug-in supports four independently-routable synthesizer objects.

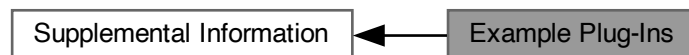
12.58.1.2.15 DemoMIDI_Sampler This simple "drum machine" style sampler plug-in demonstrates sample-accurate global and local MIDI input and the MIDI Transport interface. This plug-in uses [AAX_CMonolithicParameters](#) and therefore is AAX Native only.

12.58.1.3 Deprecated Examples

12.58.1.3.1 DemoGain_Delay

Deprecated The DemoGain_Delay example is deprecated. See DemoDelay_HostProcessor

Collaboration diagram for Example Plug-Ins:



12.59 VENUE Guide

Details about using AAX plug-ins in VENUE live sound systems.

12.59.1 Contents

- [About this document](#)
- [Overview of VENUE](#)
- [VENUE systems](#)
- [Host environment](#)
- [AAX feature support and compatibility](#)
- [VENUE Plug-in installer specification](#)
- [Additional plug-in guidelines](#)
- [System details](#)
- [Additional Information](#)

12.59.2 About this document

This guide discusses specific details related to creating AAX plug-ins which are compatible with Avid VENUE systems.

This guide includes a general overview of the new VENUE architecture as it pertains to plug-ins, a set of guidelines for developing compatible plug-ins, and details for creating full-featured plug-in installers for VENUE.

Note

Any reference in this document to "VENUE" refers specifically to VENUE | S6L, and VENUE | S3L systems. Older VENUE systems such as VENUE Profile, D-Show, and SC48 are not compatible with AAX plug-ins and are not considered in this document.

12.59.3 Overview of VENUE

VENUE is Avid's product line aimed at live sound users. VENUE systems are modular, with audio engine, control surface, console, I/O, and external GUI units.

VENUE offers plug-in racks to utilize the power of AAX DSP plug-ins. As virtual outboard racks inside the VENUE system, the plug-in racks allow users to take their AAX DSP plug-ins out of the studio and into a live performance.

Figure 1: The main VENUE software interface

Figure 2: VENUE plug-in rack

Using the VENUE GUI, an operator is able to see thumbnails for each of the plug-ins in a plug-in rack. An operator can choose to zoom in on a plug-in from this rack view and, from there, graphically control the plug-in with the mouse, keyboard, or touch-screen. Only one plug-in interface can be displayed at a time in this mode.

Figure 3: Plug-in zoom view

12.59.4 VENUE systems

This section will provide a brief overview of Avid's AAX-compatible VENUE systems. For more information about the features, functionality, and use of these systems see the VENUE user documentation.

12.59.4.1 VENUE | S6L

VENUE | S6L is a modular system designed to take on the world's most demanding tours and events with ease. Offering unprecedented processing capabilities - with over 300 processing channels - S6L delivers unrelenting performance and reliability through its advanced engine design and backs it up with modern touchscreen workflows and scalability to meet any challenge.

The S6L engine contains dedicated HDX-powered DSPs handling all plug-in processing and supports 64-bit AAX DSP plug-ins.

12.59.4.2 VENUE | S3L-X

The VENUE | S3L-X System is a modular live sound solution including an HDX-powered processing engine, scalable remote I/O, and a EUCON-enabled control surface.

At the heart of the S3L system lies the E3 engine. The E3 runs Windows Embedded and a version of VENUE software that can load AAX DSP plug-ins onto a built-in HDX platform. Accompanying the E3 engine is the [S3](#) control surface and one or more Stage 16 remote I/O boxes.

Most system parameters, including plug-ins, can be controlled using either the on-screen VENUE software or directly via encoders on the S3 control surface. When being used as part of a VENUE | S3L-X system, the S3 control surface is divided into three main sections: A - Channel Section The Channel section provides control of Input Channels, FX Returns, Output Channels, some channel parameters (such as Input Channel Gain and Aux Send levels), and channel banking. Channels are selected using the channel Select switches next to each fader.

B - Channel Control The eight Channel Control encoders provide control of processing functions for the currently selected Input or Output Channel. Inserted Dynamics and EQ plug-ins can be selected and adjusted in Channel Control.

C - Global Control The eight Global Control encoders provide control of system-wide parameters, including control of plug-ins. The Global Control encoders can be placed into Insert Mode, and can then be used to select and adjust any plug-ins.

Figure 4: Main control sections on the S3 control surface

12.59.4.2.1 Using Channel Control If a channel has an EQ, Comp/Lim, or Expander/Gate plug-in inserted on it, it can be controlled using the eight Channel Control encoders. The user can toggle between controlling the built-in Dynamics or EQ processors and the plug-in versions.

Each Input and Output Channel has built-in EQ and Comp/Lim processors. Each Input Channel also has a built-in Expander/Gate. To adjust the built-in processors, the user selects a channel, assigns a processing function to Channel Control by pressing the corresponding encoder from the Channel Control main menu, then adjusts the available parameters.

Figure 5: The Channel Control main menu

See [Center Section Parameter Mapping on VENUE | S3L-X](#) for a description of how plug-in parameters are mapped to the S3L Channel Control encoders for EQ, Compressor/Limiter, and Expander/Gate plug-ins.

12.59.5 Host environment

12.59.5.1 Audio engine

The audio engine in VENUE is based around Avid's HDX technology. Each VENUE S6L and S3L System contains a specialized HDX core card. For more information about HDX, see the [HDX DSP Guide](#). Because the VENUE architecture is so similar to HDX, AAX DSP plug-ins are cross-compatible with VENUE and most plug-ins will run seamlessly on VENUE with little or no modification.

Each VENUE system operates at a single native sample rate. VENUE supports multiple processing block sizes at this sample rate. Like in Pro Tools | HDX systems, each DSP chip in the system will only be able to load plug-ins using a single block size; plug-ins which process using different block sizes cannot be allocated to the same DSP.

12.59.5.2 Available DSP resources

The following information reflects plug-in processing abilities of VENUE systems:

- S3L-X
 - 4 TI C6727 DSP chips are available for plug-in processing
 - 40 plug-in rack slots are available
- S6L
 - All HDX DSP cards are dedicated to plug-in processing; each HDX DSP card has 18 TI C6727 DSP Chips
 - Depending on E6L engine type, 125 (E6L-144) or 200 (E6L-192) plug-in rack slots are available

12.59.5.3 Operating system

The core host software in a VENUE system is built upon Windows Embedded 8. The installation used on VENUE systems is a customized version of Windows 8 that includes only what is necessary for the VENUE software.

Core services from Windows 8 are available, such as the Win32 API, but some advanced services may not be available. Such services include MIDI, printing, video codec, .NET, etc. If your code relies on advanced Win32 APIs, or you are in doubt about specific APIs, please contact Avid for more information.

Using unavailable services may cause a plug-in to not load (e.g. if it attempts to link against DLLs that aren't included in the Windows Embedded 8 image) or to fail during run-time. Whenever possible, before using any advanced Win32 API, you should verify the availability of the service and/or handle the fact that the service might not be functional.

See the [VENUE Plug-in installer specification](#) section for more information about ensuring that all required run-time components are available to your plug-in.

12.59.5.4 Display

VENUE S6L requires that plug-in windows be restricted to a certain size. If a plug-in exceeds this size, it will overlap VENUE's GUI and may possibly be truncated. The specifications are as follows:

- S3L-X
 - Total GUI size: 1024 W x 768 H
 - Max plug-in window size (w/o sidechain support): 749 W x 617 H
 - Max plug-in window size (with sidechain support): 749 W x 565 H
- S6L
 - Total GUI size: 1920 W x 1080 H
 - Max plug-in window size (w/o sidechain support): 1436 W x 855 H
 - Max plug-in window size (with sidechain support): 1436 W x 796 H

VENUE will dispatch a [AAX_eNotificationEvent_MaxViewSizeChanged](#) notification indicating the maximum size for a plug-in's GUI. Calls to [AAX_IViewContainer::SetViewSize\(\)](#) will fail with an error if the plug-in attempts to set its view size to a larger value than the system supports, though the plug-in's initial GUI will be displayed (and possibly truncated) at its normal size before any resize requests are made.

The actual hardware and graphical acceleration available in VENUE systems is as follows:

	S3L-X	S6L
CPU model	Celeron P4500	Core i5-2510E
GPU model	Intel HD Graphics	Intel HD Graphics 3000
DirectX support	10.1	10.1
OpenGL support	2.1	3.1
OpenCL support	None	None
Shader model	4	4.1

12.59.5.5 Page tables

- VENUE S3L-X uses 'PcTL' (ProControl) page tables
- VENUE S6L uses 'Av46', a EUCON-style page table with a 4x6 knob cell configuration.

Note

In S6L, the 'FrTL' (C|24) page table is used as a fallback when 4x6 is not available. This is only a temporary solution to support legacy plug-ins. All plug-ins targeting VENUE S6L support must support the 4x6 knob cell layout and should not rely on this C|24 fallback behavior.

Page table design guidelines Primary plug-in parameters should be located on the first page in the page tables for a surface. This is especially true for the 4x6 knob cell layout used by S6L. Users should not be required to navigate between pages for the majority of common operations.

For more information about page tables, including additional guidelines for good page table design, see the [Page Table Guide](#).

12.59.5.6 Network communications

Some plug-ins may require interaction with other devices in a network. VENUE systems have two Gigabit Ethernet ports available:

1. **ECx port** Intended for connection of VNC Viewer to control the VENUE system remotely. The IP address and network mask for this port are user-configurable in the VENUE UI.
2. **AVB port** Intended for connection of all other VENUE system components, as well as a computer running Pro Tools software. This port always uses link-local addressing. Because of AVB traffic, the effective bandwidth of this port is limited to 100 Mb/s.

Note

Plug-ins must not use a significant portion of the available bandwidth on the AVB port, since it will affect mission-critical control connections of a VENUE system.

Both S3L-X and S6L systems include the Apple Bonjour service. Plug-ins may use Bonjour for interfacing with other software in the network. Plug-ins must not install their own version of Bonjour or attempt to modify the Bonjour installation on the system.

12.59.5.7 Host environment summary

	S3L-X	S6L
Operating System	Windows Embedded 8	Windows Embedded 8
Sample Rate	48 kHz	96 kHz
Max GUI size	749 W x 617 H (no sidechain) 749 W x 565 H (sidechain)	1436 W x 855 H (no sidechain) 1436 W x 796 H (sidechain)
Page table	'PcTL'	'Av46'

12.59.6 AAX feature support and compatibility

VENUE supports many of the same AAX features as Pro Tools. However, some features are not available in VENUE, and other features are managed differently between the two applications. This section describes how VENUE handles various optional AAX features.

12.59.6.1 Processing configurations

Architectures VENUE supports 64-bit AAX DSP plug-ins only. AAX Native and AAX Hybrid plug-ins are not supported. Plug-ins compiled for 32-bit processors are not supported, though they may be included in a VENUE-compatible .aaxplugin bundle alongside the plug-in's 64-bit binary.

Stem formats

- Mono plug-ins may be inserted as channel inserts on mono input strips and output busses.
- Stereo plug-ins can be inserted as channel inserts on stereo input strips and output busses.
- Greater-than-stereo formats are not supported by VENUE
- Multi-mono processing is not supported; an operator must use the stereo version of a plug-in in stereo processing locations.

Width-changing plug-ins Width Changing plug-ins are not allowed as inserts except in mix busses. Unlike in Pro Tools, a plug-in cannot change the output stem format of a strip or bus by using a mono-to-stereo plug-in on a mono track.

However, width-changing plug-ins are supported on output busses. For these, the outputs of the plug-in can either be routed back to an FX return or routed out to physical outputs. This functionality does not require any additional implementation specific to VENUE.

12.59.6.2 Presets and automation

Plug-In settings are persisted (saved & restored) the same way for Show files, snapshots & settings files, using a single method to extract settings and a single method to apply setting. The code uses the "chunk" APIs. That's similar to what Pro Tools does, except that for automation, VENUE uses exclusively snapshots (that is, VENUE does not record & playback individual control changes).

Many VENUE snapshots users are known to store settings for every Plug-In in every snapshot (or almost). This causes performance issues because settings are often fairly slow to load, making a snapshot recall last too long (sometimes 30 seconds or more!) whereas users expect a snapshot recall to take instantly. However, extremely frequently, settings are not actually changing from a snapshot to the next one (that is, from one snapshot to the next one, the vast majority of Plug-Ins contain the same settings). To mitigate this, VENUE calls [AAX_IEffectParameters::CompareActiveChunk\(\)](#) to determine whether a chunk from an incoming snapshot would result in any change to the plug-in's current settings. If not, the new chunk will not be loaded onto the plug-in. This

optimization is extremely effective, but requires that Plug-Ins implement the [CompareActiveChunk\(\)](#) method properly at any time (in particular regardless of whether the plug-in is visible or not). With VENUE, you must implement this API very well or the Plug-In may not be controllable. This optimization will affect settings application when a show is loaded, when presets are loaded and when snapshots are recalled.

All chunks are first compared one by one to the active chunks, until one is different or an error is returned. If any chunk compare fails (not equal or error returned), then all chunks are sent in sequence.

As it is the basic method for plug-in settings manipulation, it is critical that plug-ins process chunks as accurately and efficiently as possible.

Plug-In Chunks The size of a plug-in chunk cannot exceed 64KB in VENUE. If a Plug-In requires more than 64KB of chunk data total (all chunk sizes added), settings for this plug-in won't be persisted, snapshots won't work for this plug-in and users won't be able to load or save settings. If you can not meet this requirement, you should detect that you are running on VENUE and not declared the process type as it won't be usable.

12.59.6.3 Unsupported features

The following AAX features are not supported by VENUE. Plug-ins that require these features will not be compatible with VENUE systems. If your plug-ins use these features for advanced functionality but not for basic operation then you should document this restriction for VENUE users.

- Advanced audio routing VENUE does not support [Auxiliary Output Stems](#) from plug-ins.

Warning

[Description callback](#) calls to register auxiliary output stems will return an error code on VENUE systems, indicating that the host will not provide audio buffers for auxiliary output stems during processing. A plug-in must not attempt to write data into auxiliary output stem buffers which have not been provided by the host!

- Transport interface VENUE operates entirely in real-time and does not contain a timeline of pre-recorded audio. Therefore VENUE does not support the [AAX_ITransport](#) interface. VENUE will return [AAX_ERROR_UNIMPLEMENTED](#) to unsupported transport interface method calls.
- MIDI VENUE does not support MIDI routing to and from plug-in instances, and no [AAX MIDI features](#) are supported by VENUE.

12.59.7 VENUE Plug-in installer specification

To install plug-ins, the VENUE software includes a simple installation interface. To install a plug-in, the operator simply plugs in a USB Flash Drive with an installer for the plug-in and the plug-in will show up in an installer menu on the VENUE interface (shown below).

This menu will automatically list all the installable plug-ins on the drive. With the click of a button, the user can install the plug-ins onto his VENUE system.

Figure 6: The VENUE plug-in installer tab

For this custom installation to work properly, the plug-in installation USB Key must follow a certain layout. This layout is designed to be as flexible and as expandable as possible, giving the developer many options while retaining the simplicity that makes VENUE's plug-in installation appealing to the user. This layout is also designed to coexist on a drive with a Pro Tools plug-in install. The following is a detailed description of how the file hierarchy should be laid out.

12.59.7.1 Overview

The VENUE Plug-In installer specification is an extension of an AAX plug-in bundle, i.e. of the *MyPlugIn.aaxplugin* folder.

A standard .aaxplugin directory forms a basic, compatible plug-in installer for VENUE. See [.aaxplugin Directory Structure](#) for more information about this folder.

The following optional items can be added to the .aaxplugin folder to extend its functionality when used as a VENUE plug-in installer:

- License file that will need to be accepted by end user
- Pre-install action (either .bat script or executable or both)
- Post-install action (either .bat script or executable or both)
- Pre-uninstall action (either .bat script or executable or both)
- Post-uninstall action (either .bat script or executable or both)
- PACE Eden installer to update the version pre-installed on the VENUE system
- Factory presets
- Registry entries in a form of .reg files
- Program files to be placed in the system's C:\Program Files folder
- Plug-in thumbnails to be shown in the rack in VENUE Software UI

12.59.7.2 Directory structure

Here is a layout of the optional elements in the .aaxplugin plug-in installer directory:

- /Contents
 - *standard AAX plug-in contents*
- /Pace Eden
 - Setup.exe
 - Setup.bat
 - Version.txt
- /Program Files
 - ...
- /Thumbnails
 - *id1.bmp* (example: 424644204C41324131314C41.bmp)
 - *id2.bmp*
- /License.rtf or License.txt
- /Install_before.bat
- /Install_before.exe
- /Install_after.bat

- /Install_after.exe
- /SomeSettings.reg
- /Uninstall_before.bat
- /Uninstall_before.exe
- /Uninstall_after.bat
- /Uninstall_after.exe

12.59.7.3 Optional installer files

12.59.7.3.1 License terms A license stored as a file of either RTF or ASCII plain text format. The file must be located in the root folder of the installer. Depending on the text file format, the file name must be either *License.rtf* or *License.txt*.

12.59.7.3.2 Registry entries Registry settings to be applied during installation need to be stored in .reg files in the root folder of installer. All such files must be of the Windows Registry format. Particular file names does not matter.

Registry files are applied during plug-in installation.

It is important to not alter any system settings or settings of other software installed.

Note

Changes in the registry are not reverted during the plugin uninstallation.

12.59.7.3.3 Program files Files under the *Program Files* subfolder in the plug-in installer will be copied to the system's *C:\Program Files* folder. All contents of the *Program Files* subfolder are copied as-is into *C:\Program Files*, retaining the internal folder structure of nested directories.

These changes are undone when the plug-in is uninstalled.

12.59.7.3.4 Plug-in thumbnails VENUE uses thumbnail images to display plug-in GUIs in the plug-in rack while the full-size plug-in GUI is hidden.

If thumbnail images are not provided in the plug-in installer then VENUE will display a generic thumbnail image for the plug-in until it has been focused in Zoom Mode in the VENUE interface. In this case VENUE will create and cache a thumbnail image for the plug-in GUI the first time that it is focused.

The user may regenerate a thumbnail by right-clicking a rack with a plug-in and choosing "Recreate Thumbnail".

Including thumbnails in plug-in installers

A separate thumbnail should be provided in the plug-in installer for each variant supported by the plug-in, i.e. each unique AAX DSP ID triad registered by the plug-in. Use the "Recreate Thumbnail" feature to create the initial versions of your plug-in thumbnail images. Package these thumbnail images into your plug-in installer in order to guarantee that thumbnail images will be available to users immediately upon installing the plug-in.

Each thumbnail bitmap file is named after the following plug-in parameters:

1. [AAX_eProperty_ManufacturerID](#)
2. [AAX_eProperty_ProductID](#)
3. [AAX_eProperty_PluginID_Ti](#)

All of three are converted to hexadecimal representation and concatenated to form a file name that uniquely identifies a plug-in variant. For example, the Avid Channel Strip plug-in has a thumbnail file named 41564944 43685374 434D5469.bmp (no spaces).

The "Recreate Thumbnail" feature in VENUE will ensure that the generated thumbnail images use the correct file names, resolution, and image format.

12.59.7.3.5 Actions A plug-in installer may define custom actions for the following cases:

1. Pre-install action - executed when plug-in installation starts
2. Post-install action - executed when plug-in installation finishes
3. Pre-uninstall action - executed when plug-in uninstallation starts
4. Post-uninstall action - executed when plug-in uninstallation finishes

Each action is defined by either a .bat file or an Win32/64 executable file (.exe). Action files must be placed in the root folder of installer and use these file names:

1. Pre-install action - *Install_before.bat*, *Install_before.exe*
2. Post-install action - *Install_after.bat*, *Install_after.exe*
3. Pre-uninstall action - *Uninstall_before.bat*, *Uninstall_before.exe*
4. Post-uninstall action - *Uninstall_after.bat*, *Uninstall_after.exe*

If both .exe and .bat files are present for a certain action, then both are executed, with the .bat file being run before the .exe.

When executing an action, no exit code is tested. In order to report errors, the following needs to be done:

1. .bat files:

The following line should be used to report an error message from a script: `reg add HKCU\Software\Digidesign\tmp /f /v InstallResult /d "Error description"`

2. .exe files:

The error message needs to be added to Windows registry as a REG_SZ value in *HKEY_CURRENT_USER\Software\Digidesign\tmp* and named *InstallResult*.

The presence of the error message will abort a plug-in installation. Any error strings will be written to the VENUE logs so that Avid support will be able to see them. Error strings from these actions are not shown to the user.

12.59.7.3.6 PACE software installer

Warning

This functionality must not be used without prior approval from Avid. Before releasing **any** VENUE plug-in update with a bundled PACE installer you must contact Avid to confirm that the bundled installer will not cause issues for deployed VENUE systems.

VENUE allows plug-ins to install updated version of PACE iLok software immediately after the plug-in installation. In general, Avid tries to provide the latest Pace software with each VENUE software release and update. Therefore this step should not be necessary in most cases.

The PACE installer files must be located in *Pace Eden* subfolder of the installer. This folder must contain the following files:

- *Version.txt* containing a version of the *LDSvc.exe* PACE executable being installed. The version information must be in the form of *1.2.3.4*
- *Setup.exe* - the PACE installer itself.
- (optional) *Setup.bat* containing an installation script. Usually used to run PACE installer in a silent (no UI, no interaction) mode.

During installation, *Setup.bat*, if present, is run. Otherwise *Setup.exe* is executed with the following command line arguments:

```
Setup.exe /s /v"REINSTALLMODE=vamus REBOOT=ReallySuppress /qn"
```

If the version of installer is not higher than the version installed in system, the installation will not be performed.

An OS reboot is prompted in VENUE UI after PACE was installed.

12.59.7.4 Using a VENUE plug-in installer

In order to install a plug-in to the VENUE system, end user is expected to perform the following steps:

1. Download a VENUE Plug-in Installer(s) in a form of archive (Zip is suggested). Is it ok to have multiple plug-ins in one archive as soon as each plug-in is in own VENUE Plug-in Installer (i.e. in own .aaxplugin folder).
2. Unpack archive and copy installers to USB drive in the following way:
 - (a) "AAX Plug-Ins" folder must be placed in the root of USB drive.
 - (b) Each installer needs to be copied directly the "AAX Plug-Ins" folder. In the end, resulting folder structure will look like this:
3. Install plug-ins in a way described in documentation of a particular VENUE Software version.

12.59.8 Additional plug-in guidelines

12.59.8.1 General Reliability and Fault Tolerance

Since VENUE is a more "mission critical" type of application where there is no room for error during a live show, additional precautions have to be taken with respect to reliability of its various components. We have built provisions in VENUE to protect the system from catastrophic failure due to a plug-in crashing and bringing down the entire system. On top of this, extra care should be taken in developing stable software when targeting VENUE as a platform.

If a plug-in crashes, the user will be warned through a dialog. A crash brings down all plug-in processes, but audio keeps flowing through the console and through the DSPs, including the plug-ins' DSPs. All the effects continue to be effective, but their parameters can't be accessed or modified anymore (the show goes on...).

At this point, audio should be totally unaffected, even for the effect that caused the crash (assuming the crash took place in the host code, not the DSP code, of course). At the user's discretion, all plug-ins will be bypassed or muted (depending on where they are used in the system), any dependencies on the plug-ins' DSPs will be removed, the plug-ins' DSPs will be reset, and all the plug-ins will start again. When the rebuilding operation is complete, the user will be prompted to decide when he wishes the new plug-ins to be connected.

12.59.8.2 Plug-In Dialogs

Plug-ins should avoid invoking dialog windows in VENUE. We strongly suggest that any unnecessary dialog window your plug-in creates, whether at installation or instantiation, be removed. For VENUE-only plug-ins, we strongly suggest to not make use of any dialog windows.

Should you nevertheless need to make use of additional windows or dialogs, you need to make sure that they are front-most, so that they will not be hidden behind VENUE's GUI. The VENUE software will try to force your windows to be front-most, but it is safer if your plug-in enforces this in the first place.

12.59.8.3 Online Help

VENUE currently doesn't include any standardized help menu for plug-ins. We recommend that you use tooltips and other "live" help techniques similar to what plug-ins like ReVibe II, Reverb One, and Smack! use to help the user. For instance, when a user clicks on the "Side-Chain EQ" label of the Smack! Plug-In, here's what they see:

Figure 7: Tooltip help in Avid's Smack! plug-in

One of the major benefits of this technique is that it is supported across platforms and will work the same in all [AAX](#) hosts.

12.59.9 System details

12.59.9.1 External dependencies

AAX plug-ins may rely on the presence of the following items in VENUE systems:

- All VENUE systems
 - Bonjour service and library

Note

Plug-in installers are forbidden from installing over or modifying the pre-installed version of Bonjour on the VENUE system.

- VC 2005 x64 runtime
 - VC 2008 x64 runtime
 - VC 2010 x64 runtime
 - VC 2012 x64 runtime
 - VC 2013 x64 runtime
- S6L versions 5.7 and higher
 - VC 2015 x64 runtime
 - VC 2017 x64 runtime
- S6L versions 7.0 and higher
 - VC 2015-2019 x64 runtime
- S3L-X version 4.6.1 with "S3L-X Touch Support Patch" installed
 - VC 2015-2019 x64 runtime

Because VENUE does not execute standard software installers for plug-ins, Avid tries to keep VC runtime versions up to date relative to the moment of release of a particular VENUE Software version.

As of the time of this writing, Venue S3L-X systems are no longer receiving software updates and thus the S3L software will not be updated to include any additional system components beyond VC 2019. The last runtime update done for S3L-X was provided by the optional "S3L-X Touch Support Patch".

If you would like to provide compatibility with Venue host software which does not include your plug-in's required runtime libraries then we recommend statically linking your plug-in to these runtime libraries.

12.59.9.2 Environment variables

Both plug-in installers and actual plug-ins may rely on a presence of the following environment variables in a VENUE system:

- **DAEPLUGINSFOLDER** - is always set to the Installed Plug-ins location. Currently this is *C:\Program Files\Common Files\Digidesign\DAE\Plug-Ins*. Final backslash is absent.
- **JEX_HOST_TYPE** - equals "venue". If required, this may be used to provide a custom behavior of the plug-in when it's run on VENUE system.

12.59.9.3 Plug-in file locations

Installed Plug-Ins Located at *C:\Program Files\Common Files\Digidesign\DAE\Plug-Ins*

This folder is the only location used by VENUE software to instantiate a plug-in.

This location is different from the one used by Pro Tools and Media Composer for 64-bit [AAX](#) plug-ins. The only way for a plug-in to appear at that location is to be installed from VENUE Software's "Options">"Plug-Ins" page; standard plug-in installers will place the plug-in into a different directory.

Note

This location may change in future VENUE software releases. Plug-ins should not make any assumptions about the install directory and should rely on the VENUE plug-in installer to place them in the correct location.

Plug-ins available for installation

- Local: Located at *C:\Program Files\Common Files\Avid\Audio\Plug-Ins*
- On USB drive: Located at *(USB drive letter):\AAX Plug-Ins*

These locations can be chosen as sources for plug-in installation on VENUE Software's "Options">"Plug-Ins" page.

Cached plug-in installers Located at *D:\D-Show\Plug-In Installers*

Contains copies of plug-in installers installed via VENUE Software's "Options">"Plug-Ins" page.

This location can be chosen as source for plug-in installation on VENUE Software's "Options">"Plug-Ins" page under the name "Previous Installs".

Factory presets Located at *D:\D-Show\User Data\Effect Presets\Factory Presets*

Contains preset files for plug-ins, as well as for certain VENUE parameters. Presets are organized in folders.

Each subfolder corresponds to a particular preset type. Plug-in presets are named after the plug-in's name and the plug-in's `AAX_SPlugInChunkHeader::fProductID` value. For example, for an Avid Channel Strip plug-in the subfolder name is *Channel Strip [31313736]*, where "31313736" is an unsigned integer of the Channel Strip product ID.

Contents of subfolders are .tfx files of plug-in presets. Each file name will be visible to end user as a preset name.

Presets are copied into file location during a plug-in installation.

Plug-in thumbnails Located at *C:\Program Files\Digidesign\Plug-In Icons*

Contains .bmp files of plug-in thumbnails generated by VENUE Software as a result of saving current plug-in graphics into a bitmap. See [Plug-in thumbnails](#).

12.59.9.4 Installation process

12.59.9.4.1 Plug-in installation These are the steps followed by VENUE when installing a plug-in:

1. First, a VENUE plug-in installer is cached. This is done by copying a plug-in from installation source to the Cached VENUE Plug-In Installers location. All files are copied with an exception of the "Documentation" and "Pro Tools" folders.

All of the following steps are executed from the cached installer location, not from the original source location.

2. The pre-install batch script ("Install_before.bat"), if present, is executed. Execution assumes running the script without a console window.

Note

The pre-install script must not contain any installation steps, since it's executed before the plug-in license is accepted. In general, you should only use the pre-install script for pre-install checks.

3. The pre-install executable ("Install_before.exe"), if present, is executed. Execution assumes running the executable without a console window.

Note

The pre-install executable must not perform any installation steps, since it's executed before the plug-in license is accepted. In general, you should only use the pre-install script for pre-install checks.

4. A license ("License.rtf" or "License.txt"), if present, is shown to the user. If "License.rtf" is not found, "License.txt" is used. A license, if present, must be accepted by user; otherwise installation will be aborted.
5. All files of the VENUE plug-in installer are copied to the system Installed Plug-Ins location, keeping the .aaxplugin folder structure. Failure to copy any of the items results in installation being aborted.
6. If the plug-in installer contains a subfolder named "Program Files", its contents are copied into "C:\Program Files". Failure to copy any of items results in installation being aborted.
7. If the plug-in installer contains a subfolder named "Contents\Factory Presets", its contents are imported as plug-in presets. The "Factory Presets" folder must contain only valid plug-in .tfx preset files in an arbitrary folder structure. All preset files are read and copied into the system's Factory Presets location.

Note

It is important for a plug-in installer to contain only plug-in presets corresponding to plug-in being installed.

8. Plug-in thumbnails, if present, are copied from the "Thumbnails" subfolder of the installer to the system Plug-in Thumbnails location.
9. Registry files, if any, are imported. Every file with .reg extension in the root of plug-in installer is treated as a Windows Registry file and gets imported by calling

```
regedit /s "<file.reg>"
```

No error checking is performed.
10. The post-install batch script ("Install_after.bat"), if present, is executed. Execution assumes running the script without a console window.
11. The post-install executable ("Install_after.exe"), if present, is executed. Execution assumes running the executable without a console window.
12. The PACE software installer, if present, is run. If the version of the installer is not higher than the version installed in system, the installation is not performed.

When installing multiple plug-ins at once, PACE installation happens only after installing the final plug-in. VENUE will use the PACE installer with the highest available version among the installed plug-ins.

13. Plug-in installation is considered successful.

If errors occur during installation, the following happens:

1. Plug-in files are removed from the disk (see "File removal" section for details).
2. Cached plug-in installer is removed from the Cached VENUE Plug-in Installers location.

12.59.9.4.2 File removal File removal happens either in case of plug-in uninstallation or in case of a failed installation cleanup.

The following happens:

1. Plug-in files, as present in Cached VENUE Plug-in Installers location, are removed from Installed Plug-Ins location.
2. Plug-in Program Files files, as present in Cached VENUE Plug-in Installers location, are removed from Installed Plug-Ins location.

12.59.9.4.3 Plug-in uninstallation Plug-in installation process is done by VENUE Software. It removes a plug-in from the Installed Plugins location. Here's a step by step process of uninstalling plug-in:

1. The pre-uninstall batch script ("Uninstall_before.bat"), if present, is executed. Execution assumes running the script without a console window. NO return code is examined. Instead, a script is allowed to report an error string that will be visible to Avid support when examining VENUE log files. The following line should be used to write an error message from a script:

```
reg add HKCU\Software\Digidesign\tmp /f /v InstallResult /d "Error description"
```

The presence of this string means an error has occurred and a plug-in uninstallation will abort.

2. The pre-uninstall executable ("Uninstall_before.exe"), if present, is executed. Execution assumes running the executable without a console window. NO return code is examined. Instead, an executable is allowed to report an error string that will be visible to Avid support when examining VENUE log files. The error needs to be added to Windows registry as a REG_SZ value in *HKEY_CURRENT_USER\Software\Digidesign\tmp* and named *InstallResult*. The presence of this string means an error has occurred and a plug-in uninstallation will abort.
3. Plug-in files are removed. See [File removal](#) for details.

4. The post-uninstall batch script ("Install_after.bat"), if present, is executed. Execution assumes running the script without a console window. NO return code is examined. Instead, a script is allowed to report an error string that will be visible to Avid support when examining VENUE log files. The following line should be used to write an error message from a script:

```
reg add HKCU\Software\Digidesign\tmp /f /v InstallResult /d "Error description"
```

The presence of this string means an error has occurred and a plug-in uninstallation will abort.

5. The post-uninstall executable ("Install_after.exe"), if present, is executed. Execution assumes running the executable without a console window. NO return code is examined. Instead, an executable is allowed to report an error string that will be visible to Avid support when examining VENUE log files. The error needs to be added to Windows registry as a REG_SZ value in *HKEY_CURRENT_USER\Software\Digidesign\tmp* and named *InstallResult*. The presence of this string means an error has occurred and a plug-in uninstallation will abort.
6. Plug-in removal is complete.

Please note that plug-in being uninstalled is not being removed from the cache. Removal from the Cached VENUE Plug-in Installers is possible for plug-ins being not installed. In order to accomplish this, end user needs to go to VENUE Software's "Options">"Plug-Ins" page, right click on cached installer, and choose "Delete plug-in name".

12.59.10 Additional Information

12.59.10.1 Metering

For metering displays, VENUE uses dB units referenced to VENUE's nominal operating level of +4dBu. A signal at the nominal level in VENUE (i.e. registers 0dB on the VENUE meters) will, at unity gain, generate a +4dBu analog output signal (-20dBFS digital output signal).

As a result, a signal that registers +20dB on the VENUE meters will register 0dBFS on the plug-in meters. A signal at 0 dB in VENUE will be -20dBFS in the plug-in.

To map between dBFS units used in plug-ins and dB units used in VENUE the operator simply needs to add 20 to any plug-in dBFS value.

Collaboration diagram for VENUE Guide:



Chapter 13

Namespace Documentation

13.1 AAX Namespace Reference

Namespaces

- namespace [Exception](#)
AAX exception classes
- namespace [internal](#)

Enumerations

- enum [EStatusNibble](#) {
 [eStatusNibble_NoteOff](#) = 0x80 ,
 [eStatusNibble_NoteOn](#) = 0x90 ,
 [eStatusNibble_KeyPressure](#) = 0xA0 ,
 [eStatusNibble_ControlChange](#) = 0xB0 ,
 [eStatusNibble_ChannelMode](#) = 0xB0 ,
 [eStatusNibble_ProgramChange](#) = 0xC0 ,
 [eStatusNibble_ChannelPressure](#) = 0xD0 ,
 [eStatusNibble_PitchBend](#) = 0xE0 ,
 [eStatusNibble_SystemCommon](#) = 0xF0 ,
 [eStatusNibble_SystemRealTime](#) = 0xF0 }
Values for the status nibble in a MIDI packet.
- enum [EStatusByte](#) {
 [eStatusByte_SysExBegin](#) = 0xF0 ,
 [eStatusByte_MTCQuarterFrame](#) = 0xF1 ,
 [eStatusByte_SongPosition](#) = 0xF2 ,
 [eStatusByte_SongSelect](#) = 0xF3 ,
 [eStatusByte_TuneRequest](#) = 0xF6 ,
 [eStatusByte_SysExEnd](#) = 0xF7 ,
 [eStatusByte_TimingClock](#) = 0xF8 ,
 [eStatusByte_Start](#) = 0xFA ,
 [eStatusByte_Continue](#) = 0xFB ,
 [eStatusByte_Stop](#) = 0xFC ,
 [eStatusByte_ActiveSensing](#) = 0xFE ,
 [eStatusByte_Reset](#) = 0xFF }
Values for the status byte in a MIDI packet.

- enum [EChannelModeData](#) {
[eChannelModeData_AllSoundOff](#) = 120 ,
[eChannelModeData_ResetControllers](#) = 121 ,
[eChannelModeData_LocalControl](#) = 122 ,
[eChannelModeData_AllNotesOff](#) = 123 ,
[eChannelModeData_OmniOff](#) = 124 ,
[eChannelModeData_OmniOn](#) = 125 ,
[eChannelModeData_PolyOff](#) = 126 ,
[eChannelModeData_PolyOn](#) = 127 }
Values for the first data byte in a Channel Mode Message MIDI packet.
- enum [ESpecialData](#) {
[eSpecialData_AccentedClick](#) = 0x00 ,
[eSpecialData_UnaccentedClick](#) = 0x01 }
Special message data for the first data byte in a message.
- enum [ESampleRates](#) {
[e44100SampleRate](#) = 44100 ,
[e48000SampleRate](#) = 48000 ,
[e88200SampleRate](#) = 88200 ,
[e96000SampleRate](#) = 96000 ,
[e176400SampleRate](#) = 176400 ,
[e192000SampleRate](#) = 192000 }

Functions

- [std::string AsString](#) (const char *inStr)
- [const std::string & AsString](#) (const std::string &inStr)
- [const std::string & AsString](#) (const [Exception::Any](#) &inStr)
- [bool IsNoteOn](#) (const [AAX_CMidiPacket](#) *inPacket)
Returns true if inPacket is a Note On message.
- [bool IsNoteOff](#) (const [AAX_CMidiPacket](#) *inPacket)
Returns true if inPacket is a Note Off message, or a Note On message with velocity zero.
- [bool IsAllNotesOff](#) (const [AAX_CMidiPacket](#) *inPacket)
Returns true if inPacket is an All Sound Off or All Notes Off message.
- [bool IsAccentedClick](#) (const [AAX_CMidiPacket](#) *inPacket)
Returns true if inPacket is a special Pro Tools accented click message.
- [bool IsUnaccentedClick](#) (const [AAX_CMidiPacket](#) *inPacket)
Returns true if inPacket is a special Pro Tools unaccented click message.
- [bool IsClick](#) (const [AAX_CMidiPacket](#) *inPacket)
Returns true if inPacket is a special Pro Tools click message.
- [template<class T1 , class T2 > bool PageTableParameterMappingsAreEqual](#) (const T1 &inL, const T2 &inR)
- [template<class T1 , class T2 > bool PageTableParameterNameVariationsAreEqual](#) (const T1 &inL, const T2 &inR)
- [template<class T1 , class T2 > bool PageTablesAreEqual](#) (const T1 &inL, const T2 &inR)
- [template<class T > void CopyPageTable](#) (T &to, const T &from)
- [template<class T > std::vector< std::pair< int32_t, int32_t > > FindParameterMappingsInPageTable](#) (const T &inTable, [AAX_CParamID](#) inParameterID)
- [template<class T > void ClearMappedParameterByID](#) (T &ioTable, [AAX_CParamID](#) inParameterID)
- [void GetCStringOfLength](#) (char *stringOut, const char *stringIn, int32_t aMaxChars)

=====

- `int32_t Caseless_strcmp` (const char *cs, const char *ct)
- `std::string Binary2String` (uint32_t binaryValue, int32_t numBits)
- `uint32_t String2Binary` (const [AAX_IString](#) &s)
- `bool IsASCII` (char inChar)
- `bool IsFourCharASCII` (uint32_t inFourChar)
- `std::string AsStringFourChar` (uint32_t inFourChar)
- `std::string AsStringPropertyValue` ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) inPropertyValue)
- `std::string AsStringInt32` (int32_t inInt32)
- `std::string AsStringUInt32` (uint32_t inUInt32)
- `std::string AsStringIDTriad` (const [AAX_SPlugInIdentifierTriad](#) &inIDTriad)
- `std::string AsStringStemFormat` ([AAX_EStemFormat](#) inStemFormat, bool inAbbreviate=false)
- `std::string AsStringStemChannel` ([AAX_EStemFormat](#) inStemFormat, uint32_t inChannelIndex, bool inAbbreviate)
- `std::string AsStringResult` ([AAX_Result](#) inResult)
- `double SafeLog` (double aValue)

Double-precision safe log function. Returns zero for input values that are <= 0.0.
- `float SafeLogf` (float aValue)

Single-precision safe log function. Returns zero for input values that are <= 0.0.
- `AAX_CBoolean IsParameterIDEqual` ([AAX_CParamID](#) iParam1, [AAX_CParamID](#) iParam2)

Helper function to check if two parameter IDs are equivalent.
- `AAX_CBoolean IsEffectIDEqual` (const [AAX_IString](#) *iEffectID1, const [AAX_IString](#) *iEffectID2)

Helper function to check if two Effect IDs are equivalent.
- `AAX_CBoolean IsAvidNotification` ([AAX_CTypeID](#) inNotificationID)

Helper function to check if a notification ID is reserved for host notifications.
- `void alignFree` (void *p)
- `template<class T >`
`T * alignMalloc` (int iArraySize, int iAlignment)
- `void DeDenormal` (double &iValue)

Clamps very small floating point values to zero.
- `void DeDenormal` (float &iValue)

Clamps very small floating point values to zero.
- `void DeDenormalFine` (float &iValue)
- `void FilterDenormals` (float *inSamples, int32_t inLength)

Round all denormal/subnormal samples in a buffer to zero.
- `template<class GFLOAT >`
`GFLOAT ClampToZero` (GFLOAT iValue, GFLOAT iClampThreshold)
- `void ZeroMemorySW` (void *iPointer, int iNumBytes)
- `void ZeroMemoryDW` (void *iPointer, int iNumBytes)
- `template<typename T, int N>`
`void Fill` (T *iArray, const T *iVal)
- `template<typename T, int M, int N>`
`void Fill` (T *iArray, const T *iVal)
- `template<typename T, int L, int M, int N>`
`void Fill` (T *iArray, const T *iVal)
- `double fabs` (double iVal)
- `float fabs` (float iVal)
- `float fabsf` (float iVal)
- `template<class T >`
`T AbsMax` (const T &iValue, const T &iMax)
- `template<class T >`
`T MinMax` (const T &iValue, const T &iMin, const T &iMax)
- `template<class T >`
`T Max` (const T &iValue1, const T &iValue2)

- template<class T >
T [Min](#) (const T &iValue1, const T &iValue2)
- template<class T >
T [Sign](#) (const T &iValue)
- double [PolyEval](#) (double x, const double *coefs, int numCoefs)
- double [CeilLog2](#) (double iValue)
- void [SinCosMix](#) (float aLinearMix, float &aSinMix, float &aCosMix)
- int32_t [FastRound2Int32](#) (double iVal)
Round to Int32.
- int32_t [FastRound2Int32](#) (float iVal)
Round to Int32.
- int32_t [FastRndDbI2Int32](#) (double iVal)
- int32_t [FastTrunc2Int32](#) (double iVal)
Float to Int conversion with truncation.
- int32_t [FastTrunc2Int32](#) (float iVal)
Float to Int conversion with truncation.
- int64_t [FastRound2Int64](#) (double iVal)
Round to Int64.
- int32_t [GetInt32RPDF](#) (int32_t *iSeed)
- int32_t [GetFastInt32RPDF](#) (int32_t *iSeed)
CALL: Calculate pseudo-random 32 bit number based on linear congruential method.
- float [GetRPDFWithAmplitudeOneHalf](#) (int32_t *iSeed)
- float [GetRPDFWithAmplitudeOne](#) (int32_t *iSeed)
- float [GetFastRPDFWithAmplitudeOne](#) (int32_t *iSeed)
- float [GetTPDFWithAmplitudeOne](#) (int32_t *iSeed)

MIDI logging utilities

- void [AsStringMIDIStream_Debug](#) (const [AAX_CMidiStream](#) &inStream, char *outBuffer, int32_t inBuffer↵
Size)

Filesystem utilities

- bool [GetPathToPlugInBundle](#) (const char *iBundleName, int iMaxLength, char *oModuleName)
Retrieve the file path of the .aaxplugin bundle.

Variables

- const int [cBigEndian](#) =0
- const int [cLittleEndian](#) =1
- const double [cPi](#) = 3.1415926535897932384626433832795
- const double [cTwoPi](#) = 6.2831853071795862319959269370884
- const double [cHalfPi](#) = 1.5707963267948965579989817342721
- const double [cQuarterPi](#) = 0.78539816339744827899949086713605
- const double [cRootTwo](#) = 1.4142135623730950488016887242097
- const double [cOneOverRootTwo](#) = 0.70710678118654752440084436210485
- const double [cPos3dB](#) =1.4142135623730950488016887242097
- const double [cNeg3dB](#) =0.70710678118654752440084436210485
- const double [cPos6dB](#) =2.0
- const double [cNeg6dB](#) =0.5
- const double [cNormalizeLongToAmplitudeOneHalf](#) = 0.00000000023283064365386962890625
- const double [cNormalizeLongToAmplitudeOne](#) = 1.0/double(1<<31)

- const double `cMilli` =0.001
- const double `cMicro` =0.001*0.001
- const double `cNano` =0.001*0.001*0.001
- const double `cPico` =0.001*0.001*0.001*0.001
- const double `cKilo` =1000.0
- const double `cMega` =1000.0*1000.0
- const double `cGiga` =1000.0*1000.0*1000.0
- const double `cDenormalAvoidanceOffset` =3.0e-34
- const float `cFloatDenormalAvoidanceOffset` =3.0e-20f
- const unsigned int `kPowExtent` = 9
- const unsigned int `kPowTableSize` = 1 << `kPowExtent`
- const float `cSeedDivisor` = 1/127773.0f
- const int32_t `clInitialSeedValue` =0x00F54321

13.1.1 Enumeration Type Documentation

13.1.1.1 EStatusNibble

enum `AAX::EStatusNibble`

Values for the status nibble in a MIDI packet.

Enumerator

<code>eStatusNibble_NoteOff</code>	
<code>eStatusNibble_NoteOn</code>	
<code>eStatusNibble_KeyPressure</code>	
<code>eStatusNibble_ControlChange</code>	
<code>eStatusNibble_ChannelMode</code>	
<code>eStatusNibble_ProgramChange</code>	
<code>eStatusNibble_ChannelPressure</code>	
<code>eStatusNibble_PitchBend</code>	
<code>eStatusNibble_SystemCommon</code>	
<code>eStatusNibble_SystemRealTime</code>	

13.1.1.2 EStatusByte

enum `AAX::EStatusByte`

Values for the status byte in a MIDI packet.

Enumerator

<code>eStatusByte_SysExBegin</code>	
-------------------------------------	--

Enumerator

eStatusByte_MTCQuarterFrame	
eStatusByte_SongPosition	
eStatusByte_SongSelect	
eStatusByte_TuneRequest	
eStatusByte_SysExEnd	
eStatusByte_TimingClock	
eStatusByte_Start	
eStatusByte_Continue	
eStatusByte_Stop	
eStatusByte_ActiveSensing	
eStatusByte_Reset	

13.1.1.3 EChannelModeData

enum [AAX::EChannelModeData](#)

Values for the first data byte in a Channel Mode Message MIDI packet.

Enumerator

eChannelModeData_AllSoundOff	
eChannelModeData_ResetControllers	
eChannelModeData_LocalControl	
eChannelModeData_AllNotesOff	
eChannelModeData_OmniOff	
eChannelModeData_OmniOn	
eChannelModeData_PolyOff	
eChannelModeData_PolyOn	

13.1.1.4 ESpecialData

enum [AAX::ESpecialData](#)

Special message data for the first data byte in a message.

Enumerator

eSpecialData_AccentedClick	For use when the high status nibble is eStatusNibble_NoteOn and the low status nibble is zero.
eSpecialData_UnaccentedClick	For use when the high status nibble is eStatusNibble_NoteOn and the low status nibble is zero.

13.1.1.5 ESsampleRates

enum [AAX::ESampleRates](#)

Enumerator

e44100SampleRate	
e48000SampleRate	
e88200SampleRate	
e96000SampleRate	
e176400SampleRate	
e192000SampleRate	

13.1.2 Function Documentation

13.1.2.1 AsString() [1/3]

```
std::string AAX::AsString (
    const char * inStr ) [inline]
```

Generic conversion of a string-like object to a std::string

13.1.2.2 AsString() [2/3]

```
const std::string & AAX::AsString (
    const std::string & inStr ) [inline]
```

Generic conversion of a string-like object to a std::string

13.1.2.3 AsString() [3/3]

```
const std::string & AAX::AsString (
    const Exception::Any & inStr ) [inline]
```

Generic conversion of a string-like object to a std::string

References [AAX::Exception::Any::What\(\)](#).

Here is the call graph for this function:



13.1.2.4 IsNoteOn()

```
bool AAX::IsNoteOn (
    const AAX\_CMidiPacket * inPacket ) [inline]
```

Returns true if *inPacket* is a Note On message.

References [eStatusNibble_NoteOn](#), and [AAX_CMidiPacket::mData](#).

13.1.2.5 IsNoteOff()

```
bool AAX::IsNoteOff (
    const AAX\_CMidiPacket * inPacket ) [inline]
```

Returns true if *inPacket* is a Note Off message, or a Note On message with velocity zero.

References [eStatusNibble_NoteOff](#), [eStatusNibble_NoteOn](#), and [AAX_CMidiPacket::mData](#).

13.1.2.6 IsAllNotesOff()

```
bool AAX::IsAllNotesOff (
    const AAX\_CMidiPacket * inPacket ) [inline]
```

Returns true if *inPacket* is an All Sound Off or All Notes Off message.

References [eChannelModeData_AllNotesOff](#), [eChannelModeData_AllSoundOff](#), [eChannelModeData_OmniOff](#), [eChannelModeData_OmniOn](#), [eChannelModeData_PolyOff](#), [eChannelModeData_PolyOn](#), [eStatusNibble_ChannelMode](#), and [AAX_CMidiPacket::mData](#).

13.1.2.7 IsAccentedClick()

```
bool AAX::IsAccentedClick (
    const AAX_CMidiPacket * inPacket ) [inline]
```

Returns true if `inPacket` is a special Pro Tools accented click message.

References [eSpecialData_AccentedClick](#), [eStatusNibble_NoteOn](#), and [AAX_CMidiPacket::mData](#).

Referenced by [IsClick\(\)](#).

Here is the caller graph for this function:



13.1.2.8 IsUnaccentedClick()

```
bool AAX::IsUnaccentedClick (
    const AAX_CMidiPacket * inPacket ) [inline]
```

Returns true if `inPacket` is a special Pro Tools unaccented click message.

References [eSpecialData_UnaccentedClick](#), [eStatusNibble_NoteOn](#), and [AAX_CMidiPacket::mData](#).

Referenced by [IsClick\(\)](#).

Here is the caller graph for this function:



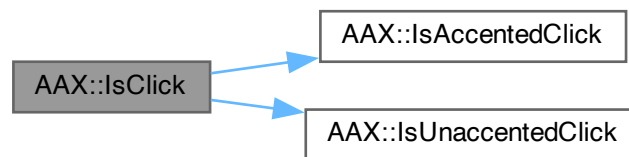
13.1.2.9 IsClick()

```
bool AAX::IsClick (
    const AAX_CMidiPacket * inPacket ) [inline]
```

Returns true if `inPacket` is a special Pro Tools click message.

References [IsAccentedClick\(\)](#), and [IsUnaccentedClick\(\)](#).

Here is the call graph for this function:



13.1.2.10 PageTableParameterMappingsAreEqual()

```
template<class T1 , class T2 >
bool AAX::PageTableParameterMappingsAreEqual (
    const T1 & inL,
    const T2 & inR ) [inline]
```

Compare the parameter mappings in two page tables

T1 and T2 : Page table class types (e.g. [AAX_IACFPPageTable](#), [AAX_IPageTable](#))

References [AAX_SUCCESS](#).

Referenced by [PageTablesAreEqual\(\)](#).

Here is the caller graph for this function:



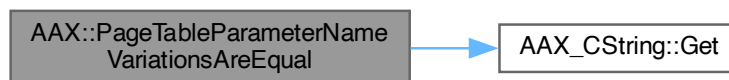
13.1.2.11 PageTableParameterNameVariationsAreEqual()

```
template<class T1 , class T2 >
bool AAX::PageTableParameterNameVariationsAreEqual (
    const T1 & inL,
    const T2 & inR ) [inline]
```

References [AAX_SUCCESS](#), and [AAX_CString::Get\(\)](#).

Referenced by [PageTablesAreEqual\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

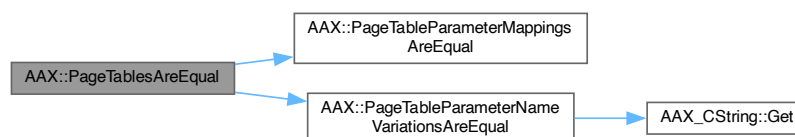


13.1.2.12 PageTablesAreEqual()

```
template<class T1 , class T2 >
bool AAX::PageTablesAreEqual (
    const T1 & inL,
    const T2 & inR ) [inline]
```

References [PageTableParameterMappingsAreEqual\(\)](#), and [PageTableParameterNameVariationsAreEqual\(\)](#).

Here is the call graph for this function:



13.1.2.13 CopyPageTable()

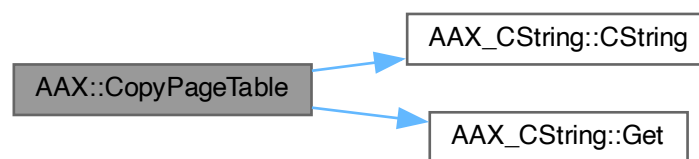
```
template<class T >
void AAX::CopyPageTable (
    T & to,
    const T & from ) [inline]
```

Copy a page table

T: A page table class type (e.g. [AAX_IACFPPageTable](#), [AAX_IPageTable](#))

References [AAX_SUCCESS](#), [AAX_CString::CString\(\)](#), and [AAX_CString::Get\(\)](#).

Here is the call graph for this function:



13.1.2.14 FindParameterMappingsInPageTable()

```
template<class T >
std::vector< std::pair< int32_t, int32_t > > AAX::FindParameterMappingsInPageTable (
    const T & inTable,
    AAX_CParamID inParameterID ) [inline]
```

Find all slots where a particular parameter is mapped

T: A page table class type (e.g. [AAX_IACFPPageTable](#), [AAX_IPageTable](#))

Returns

A vector of pairs of [page index, slot index] each representing a single mapping of the parameter

References [AAX_SUCCESS](#).

Referenced by [ClearMappedParameterByID\(\)](#).

Here is the caller graph for this function:



13.1.2.15 ClearMappedParameterByID()

```
template<class T >
void AAX::ClearMappedParameterByID (
    T & ioTable,
    AAX_CParamID inParameterID ) [inline]
```

Remove all mappings of a particular from a page table

T : A page table class type (e.g. [AAX_IACFPageTable](#), [AAX_IPageTable](#))

References [FindParameterMappingsInPageTable\(\)](#).

Here is the call graph for this function:



13.1.2.16 GetCStringOfLength()

```
void AAX::GetCStringOfLength (
    char * stringOut,
    const char * stringIn,
    int32_t aMaxChars ) [inline]
```

=====

References [AAX_ASSERT](#).

13.1.2.17 Caseless_strcmp()

```
int32_t AAX::Caseless_strcmp (
    const char * cs,
    const char * ct ) [inline]
```

13.1.2.18 Binary2String()

```
std::string AAX::Binary2String (
    uint32_t binaryValue,
    int32_t numBits ) [inline]
```

Referenced by [AsStringPropertyValue\(\)](#).

Here is the caller graph for this function:

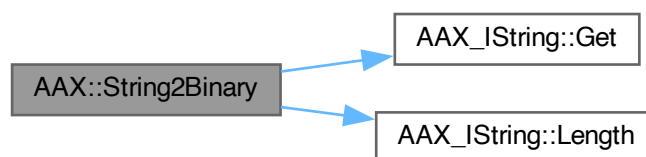


13.1.2.19 String2Binary()

```
uint32_t AAX::String2Binary (
    const AAX\_IString & s ) [inline]
```

References [AAX_ASSERT](#), [AAX_IString::Get\(\)](#), and [AAX_IString::Length\(\)](#).

Here is the call graph for this function:



13.1.2.20 IsASCII()

```
bool AAX::IsASCII (
    char inChar ) [inline]
```

Referenced by [AsStringFourChar\(\)](#), and [IsFourCharASCII\(\)](#).

Here is the caller graph for this function:



13.1.2.21 IsFourCharASCII()

```
bool AAX::IsFourCharASCII (
    uint32_t inFourChar ) [inline]
```

References [IsASCII\(\)](#).

Referenced by [AsStringPropertyValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



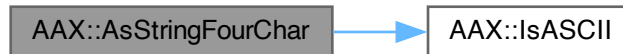
13.1.2.22 AsStringFourChar()

```
std::string AAX::AsStringFourChar (
    uint32_t inFourChar ) [inline]
```

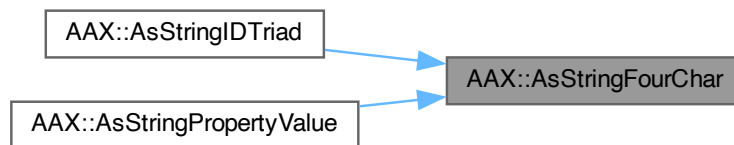
References [AAX_CONSTEXPR](#), and [IsASCII\(\)](#).

Referenced by [AsStringIDTriad\(\)](#), and [AsStringPropertyValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

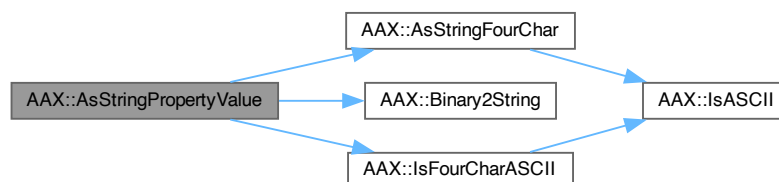


13.1.2.23 AsStringPropertyValue()

```
std::string AAX::AsStringPropertyValue (
    AAX_EProperty inProperty,
    AAX_CPropertyValue inPropertyValue ) [inline]
```

References [AAX_CONSTEXPR](#), [AAX_eProperty_Constraint_Location](#), [AAX_eProperty_SampleRate](#), [AsStringFourChar\(\)](#), [Binary2String\(\)](#), and [IsFourCharASCII\(\)](#).

Here is the call graph for this function:

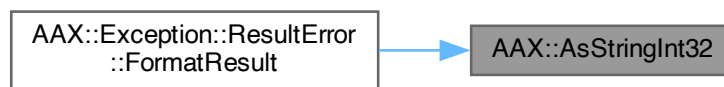


13.1.2.24 AsStringInt32()

```
std::string AAX::AsStringInt32 (
    int32_t inInt32 ) [inline]
```

Referenced by [AAX::Exception::ResultError::FormatResult\(\)](#).

Here is the caller graph for this function:



13.1.2.25 AsStringUInt32()

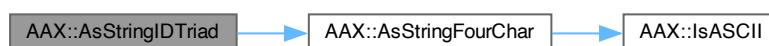
```
std::string AAX::AsStringUInt32 (
    uint32_t inUInt32 ) [inline]
```

13.1.2.26 AsStringIDTriad()

```
std::string AAX::AsStringIDTriad (
    const AAX_SPlugInIdentifierTriad & inIDTriad ) [inline]
```

References [AsStringFourChar\(\)](#), [AAX_SPlugInIdentifierTriad::mManufacturerID](#), [AAX_SPlugInIdentifierTriad::mPlugInID](#), and [AAX_SPlugInIdentifierTriad::mProductID](#).

Here is the call graph for this function:



13.1.2.27 AsStringStemFormat()

```
std::string AAX::AsStringStemFormat (
    AAX_eStemFormat inStemFormat,
    bool inAbbreviate = false ) [inline]
```

References [AAX_eStemFormat_5_0](#), [AAX_eStemFormat_5_0_2](#), [AAX_eStemFormat_5_0_4](#), [AAX_eStemFormat_5_1](#), [AAX_eStemFormat_5_1_2](#), [AAX_eStemFormat_5_1_4](#), [AAX_eStemFormat_6_0](#), [AAX_eStemFormat_6_1](#), [AAX_eStemFormat_7_0_2](#), [AAX_eStemFormat_7_0_4](#), [AAX_eStemFormat_7_0_6](#), [AAX_eStemFormat_7_0_DTS](#), [AAX_eStemFormat_7_0_SDDS](#), [AAX_eStemFormat_7_1_2](#), [AAX_eStemFormat_7_1_4](#), [AAX_eStemFormat_7_1_6](#), [AAX_eStemFormat_7_1_DTS](#), [AAX_eStemFormat_7_1_SDDS](#), [AAX_eStemFormat_9_0_4](#), [AAX_eStemFormat_9_0_6](#), [AAX_eStemFormat_9_1_4](#), [AAX_eStemFormat_9_1_6](#), [AAX_eStemFormat_Ambi_1_ACN](#), [AAX_eStemFormat_Ambi_2_ACN](#), [AAX_eStemFormat_Ambi_3_ACN](#), [AAX_eStemFormat_Ambi_4_ACN](#), [AAX_eStemFormat_Ambi_5_ACN](#), [AAX_eStemFormat_Ambi_6_ACN](#), [AAX_eStemFormat_Ambi_7_ACN](#), [AAX_eStemFormat_Any](#), [AAX_eStemFormat_INT32_MAX](#), [AAX_eStemFormat_LCR](#), [AAX_eStemFormat_LCRS](#), [AAX_eStemFormat_Mono](#), [AAX_eStemFormat_None](#), [AAX_eStemFormat_Quad](#), [AAX_eStemFormat_Stereo](#), and [AAX_eStemFormatNum](#).

13.1.2.28 AsStringStemChannel()

```
std::string AAX::AsStringStemChannel (
    AAX_eStemFormat inStemFormat,
    uint32_t inChannelIndex,
    bool inAbbreviate ) [inline]
```

References [AAX_eStemFormat_5_0](#), [AAX_eStemFormat_5_0_2](#), [AAX_eStemFormat_5_0_4](#), [AAX_eStemFormat_5_1](#), [AAX_eStemFormat_5_1_2](#), [AAX_eStemFormat_5_1_4](#), [AAX_eStemFormat_6_0](#), [AAX_eStemFormat_6_1](#), [AAX_eStemFormat_7_0_2](#), [AAX_eStemFormat_7_0_4](#), [AAX_eStemFormat_7_0_6](#), [AAX_eStemFormat_7_0_DTS](#), [AAX_eStemFormat_7_0_SDDS](#), [AAX_eStemFormat_7_1_2](#), [AAX_eStemFormat_7_1_4](#), [AAX_eStemFormat_7_1_6](#), [AAX_eStemFormat_7_1_DTS](#), [AAX_eStemFormat_7_1_SDDS](#), [AAX_eStemFormat_9_0_4](#), [AAX_eStemFormat_9_0_6](#), [AAX_eStemFormat_9_1_4](#), [AAX_eStemFormat_9_1_6](#), [AAX_eStemFormat_Ambi_1_ACN](#), [AAX_eStemFormat_Ambi_2_ACN](#), [AAX_eStemFormat_Ambi_3_ACN](#), [AAX_eStemFormat_Ambi_4_ACN](#), [AAX_eStemFormat_Ambi_5_ACN](#), [AAX_eStemFormat_Ambi_6_ACN](#), [AAX_eStemFormat_Ambi_7_ACN](#), [AAX_eStemFormat_Any](#), [AAX_eStemFormat_INT32_MAX](#), [AAX_eStemFormat_LCR](#), [AAX_eStemFormat_LCRS](#), [AAX_eStemFormat_Mono](#), [AAX_eStemFormat_None](#), [AAX_eStemFormat_Quad](#), [AAX_eStemFormat_Stereo](#), and [AAX_eStemFormatNum](#).

13.1.2.29 AsStringResult()

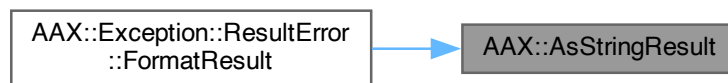
```
std::string AAX::AsStringResult (
    AAX_Result inResult ) [inline]
```

References [AAX_ERROR_ACF_ERROR](#), [AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW](#), [AAX_ERROR_CONTEXT_ALREADY](#), [AAX_ERROR_DIRECT_ACCESS_OUT_OF_BOUNDS](#), [AAX_ERROR_DUPLICATE_EFFECT_ID](#), [AAX_ERROR_DUPLICATE_TYP](#), [AAX_ERROR_EMPTY_EFFECT_NAME](#), [AAX_ERROR_FIELD_TYPE_DOES_NOT_SUPPORT_DIRECT_ACCESS](#), [AAX_ERROR_FIFO_FULL](#), [AAX_ERROR_INCORRECT_CHUNK_SIZE](#), [AAX_ERROR_INITIALIZING_PACKET_STREAM_THREA](#), [AAX_ERROR_INVALID_ARGUMENT](#), [AAX_ERROR_INVALID_CHUNK_ID](#), [AAX_ERROR_INVALID_CHUNK_INDEX](#), [AAX_ERROR_INVALID_FIELD_INDEX](#), [AAX_ERROR_INVALID_INTERNAL_DATA](#), [AAX_ERROR_INVALID_METER_INDEX](#), [AAX_ERROR_INVALID_METER_TYPE](#), [AAX_ERROR_INVALID_PARAMETER_ID](#), [AAX_ERROR_INVALID_PARAMETER_INDEX](#), [AAX_ERROR_INVALID_PATH](#), [AAX_ERROR_INVALID_STRING_CONVERSION](#), [AAX_ERROR_INVALID_VIEW_SIZE](#), [AAX_ERROR_MALFORMED_CHUNK](#), [AAX_ERROR_MIXER_THREAD_FALLING_BEHIND](#), [AAX_ERROR_NO_COMPONENTS](#), [AAX_ERROR_NOT_INITIALIZED](#), [AAX_ERROR_NOTIFICATION_FAILED](#), [AAX_ERROR_NULL_ARGUMENT](#),

[AAX_ERROR_NULL_COMPONENT](#), [AAX_ERROR_NULL_OBJECT](#), [AAX_ERROR_OLDER_VERSION](#), [AAX_ERROR_PLUGIN_BEGIN](#), [AAX_ERROR_PLUGIN_END](#), [AAX_ERROR_PLUGIN_NOT_AUTHORIZED](#), [AAX_ERROR_PLUGIN_NULL_PARAMETER](#), [AAX_ERROR_PORT_ID_OUT_OF_RANGE](#), [AAX_ERROR_POST_PACKET_FAILED](#), [AAX_ERROR_PROPERTY_UNDEFINED](#), [AAX_ERROR_SIGNED_INT_OVERFLOW](#), [AAX_ERROR_TOD_BEHIND](#), [AAX_ERROR_UNIMPLEMENTED](#), [AAX_ERROR_UNKNOWN_EXCEPTION](#), [AAX_ERROR_UNKNOWN_ID](#), [AAX_ERROR_UNKNOWN_PLUGIN](#), [AAX_RESULT_ADD_FIELD_UNSUPPORTED_FIELD_TYPE](#), [AAX_RESULT_NEW_PACKET](#), [AAX_RESULT_PACKET_STREAM_NOT_EMPTY](#), [AAX_SUCCESS](#), and [DEFINE_AAX_ERROR_STRING](#).

Referenced by [AAX::Exception::ResultError::FormatResult\(\)](#).

Here is the caller graph for this function:



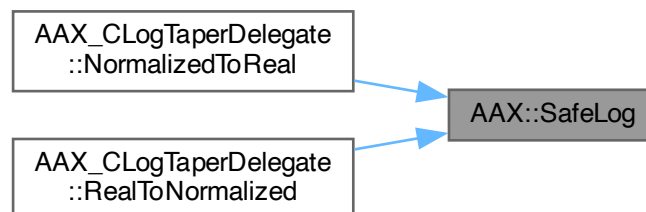
13.1.2.30 SafeLog()

```
double AAX::SafeLog (
    double aValue ) [inline]
```

Double-precision safe log function. Returns zero for input values that are ≤ 0.0 .

Referenced by [AAX_CLogTaperDelegate< T, RealPrecision >::NormalizedToReal\(\)](#), and [AAX_CLogTaperDelegate< T, RealPrecision >::RealToNormalized\(\)](#).

Here is the caller graph for this function:



13.1.2.31 SafeLogf()

```
float AAX::SafeLogf (
    float aValue ) [inline]
```

Single-precision safe log function. Returns zero for input values that are ≤ 0.0 .

13.1.2.32 IsParameterIDEqual()

```
AAX_CBoolean AAX::IsParameterIDEqual (
    AAX_CParamID iParam1,
    AAX_CParamID iParam2 ) [inline]
```

Helper function to check if two parameter IDs are equivalent.

13.1.2.33 IsEffectIDEqual()

```
AAX_CBoolean AAX::IsEffectIDEqual (
    const AAX_IString * iEffectID1,
    const AAX_IString * iEffectID2 ) [inline]
```

Helper function to check if two Effect IDs are equivalent.

References [AAX_IString::Get\(\)](#).

Here is the call graph for this function:



13.1.2.34 IsAvidNotification()

```
AAX_CBoolean AAX::IsAvidNotification (
    AAX_CTypeID inNotificationID ) [inline]
```

Helper function to check if a notification ID is reserved for host notifications.

13.1.2.35 alignFree()

```
void AAX::alignFree (
    void * p ) [inline]
```

13.1.2.36 alignMalloc()

```
template<class T >
T * AAX::alignMalloc (
    int iArraySize,
    int iAlignment )
```

13.1.2.37 DeDenormal() [1/2]

```
void AAX::DeDenormal (
    double & iValue ) [inline]
```

Clamps very small floating point values to zero.

On Pentiums and Pentium IIs the generation of denormal floats causes enormous performance losses. This routine removes denormals by clamping very small values to zero. The clamping threshold is very small, but is not the absolute minimum. If absolute minimum clamping is desired, use [AAX::DeDenormalFine\(\)](#)

References [cDenormalAvoidanceOffset](#).

13.1.2.38 DeDenormal() [2/2]

```
void AAX::DeDenormal (
    float & iValue ) [inline]
```

Clamps very small floating point values to zero.

On Pentiums and Pentium IIs the generation of denormal floats causes enormous performance losses. This routine removes denormals by clamping very small values to zero. The clamping threshold is very small, but is not the absolute minimum. If absolute minimum clamping is desired, use [AAX::DeDenormalFine\(\)](#)

References [cFloatDenormalAvoidanceOffset](#).

13.1.2.39 DeDenormalFine()

```
void AAX::DeDenormalFine (
    float & iValue ) [inline]
```

Similar to [AAX::DeDenormal\(\)](#), but uses the minimum possible normal float value as the clamping threshold

13.1.2.40 FilterDenormals()

```
void AAX::FilterDenormals (
    float * inSamples,
    int32_t inLength ) [inline]
```

Round all denormal/subnormal samples in a buffer to zero.

Parameters

in	<i>inSamples</i>	Samples to convert
in	<i>inLength</i>	Number of samples in inSamples

References [fabsf\(\)](#).

Here is the call graph for this function:



13.1.2.41 ClampToZero()

```

template<class GFLOAT >
GFLOAT AAX::ClampToZero (
    GFLOAT iValue,
    GFLOAT iClampThreshold ) [inline]
  
```

13.1.2.42 ZeroMemorySW()

```

void AAX::ZeroMemorySW (
    void * iPointer,
    int iNumBytes ) [inline]
  
```

13.1.2.43 ZeroMemoryDW()

```

void AAX::ZeroMemoryDW (
    void * iPointer,
    int iNumBytes ) [inline]
  
```

13.1.2.44 Fill() [1/3]

```
template<typename T , int N>  
void AAX::Fill (  
    T * iArray,  
    const T * iVal )
```

Referenced by [Fill\(\)](#).

Here is the caller graph for this function:

**13.1.2.45 Fill()** [2/3]

```
template<typename T , int M, int N>  
void AAX::Fill (  
    T * iArray,  
    const T * iVal ) [inline]
```

References [Fill\(\)](#).

Here is the call graph for this function:



13.1.2.46 Fill() [3/3]

```
template<typename T , int L, int M, int N>  
void AAX::Fill (  
    T * iArray,  
    const T * iVal ) [inline]
```

References [Fill\(\)](#).

Here is the call graph for this function:



13.1.2.47 fabs() [1/2]

```
double AAX::fabs (  
    double iVal ) [inline]
```

Referenced by [fabsf\(\)](#).

Here is the caller graph for this function:



13.1.2.48 fabs() [2/2]

```
float AAX::fabs (  
    float iVal ) [inline]
```

13.1.2.49 fabsf()

```
float AAX::fabsf (
    float iVal ) [inline]
```

References [fabs\(\)](#).

Referenced by [FilterDenormals\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.2.50 AbsMax()

```
template<class T >
T AAX::AbsMax (
    const T & iValue,
    const T & iMax ) [inline]
```

13.1.2.51 MinMax()

```
template<class T >
T AAX::MinMax (
    const T & iValue,
    const T & iMin,
    const T & iMax ) [inline]
```

13.1.2.52 Max()

```
template<class T >
T AAX::Max (
    const T & iValue1,
    const T & iValue2 ) [inline]
```

13.1.2.53 Min()

```
template<class T >
T AAX::Min (
    const T & iValue1,
    const T & iValue2 ) [inline]
```

13.1.2.54 Sign()

```
template<class T >
T AAX::Sign (
    const T & iValue ) [inline]
```

13.1.2.55 PolyEval()

```
double AAX::PolyEval (
    double x,
    const double * coefs,
    int numCoefs ) [inline]
```

13.1.2.56 CeilLog2()

```
double AAX::CeilLog2 (
    double iValue ) [inline]
```

13.1.2.57 SinCosMix()

```
void AAX::SinCosMix (
    float aLinearMix,
    float & aSinMix,
    float & aCosMix ) [inline]
```

References [cHalfPi](#).

13.1.2.58 FastRound2Int32() [1/2]

```
int32_t AAX::FastRound2Int32 (
    double iVal ) [inline]
```

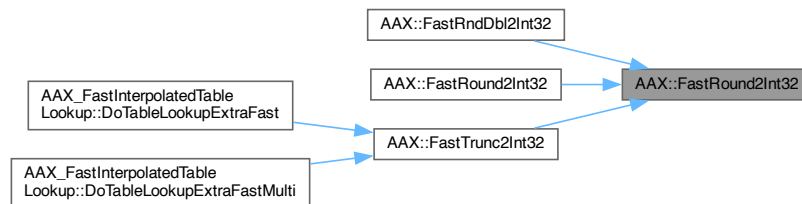
Round to Int32.

Parameters

in	<i>iVal</i>	Value to convert
----	-------------	------------------

Referenced by [FastRndDbl2Int32\(\)](#), [FastRound2Int32\(\)](#), and [FastTrunc2Int32\(\)](#).

Here is the caller graph for this function:



13.1.2.59 FastRound2Int32() [2/2]

```
int32_t AAX::FastRound2Int32 (
    float iVal ) [inline]
```

Round to Int32.

Parameters

in	<i>iVal</i>	Value to convert
----	-------------	------------------

References [FastRound2Int32\(\)](#).

Here is the call graph for this function:



13.1.2.60 FastRndDbl2Int32()

```
int32_t AAX::FastRndDbl2Int32 (
    double iVal ) [inline]
```

Deprecated

References [FastRound2Int32\(\)](#).

Here is the call graph for this function:

**13.1.2.61 FastTrunc2Int32()** [1/2]

```
int32_t AAX::FastTrunc2Int32 (
    double iVal ) [inline]
```

Float to Int conversion with truncation.

Parameters

in	<i>iVal</i>	Value to convert
----	-------------	------------------

Note

This truncation is NOT identical to C style casting. Because the Intel (and I would assume PowerPC) processors use convergent rounding by default, exactly whole odd numbers will truncate down by 1.0 (e.g. 0.0->0, 1.0->0, 2.0->2, 3.0->2). Surprisingly, even with these limitations this fast float to int conversion is often very useful in practice, as long as one is aware of these issues.

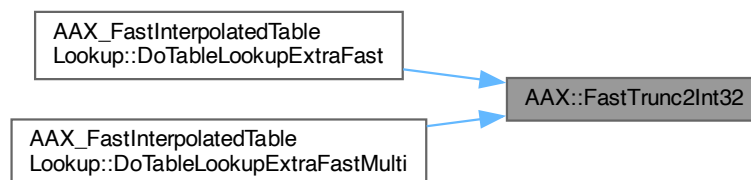
References [FastRound2Int32\(\)](#).

Referenced by [AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::DoTableLookupExtraFast\(\)](#), and [AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::DoTableLookupExtraFastMulti\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.2.62 FastTrunc2Int32() [2/2]

```
int32_t AAX::FastTrunc2Int32 (
    float iVal ) [inline]
```

Float to Int conversion with truncation.

Parameters

in	<i>iVal</i>	Value to convert
----	-------------	------------------

13.1.2.63 FastRound2Int64()

```
int64_t AAX::FastRound2Int64 (
    double iVal ) [inline]
```

Round to Int64.

Taken from Paul V's implementation in `Sys_VecUtils`. This only works on values smaller than 2^{52} .

Parameters

<i>in</i>	<i>iVal</i>	Value to convert
-----------	-------------	------------------

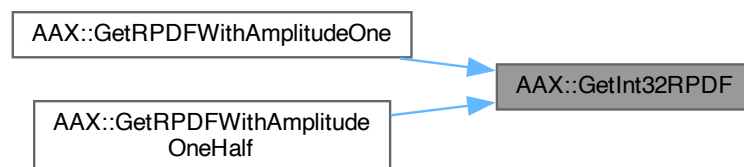
13.1.2.64 GetInt32RPDF()

```
int32_t AAX::GetInt32RPDF (
    int32_t * iSeed ) [inline]
```

References [cSeedDivisor](#).

Referenced by [GetRPDFWithAmplitudeOne\(\)](#), and [GetRPDFWithAmplitudeOneHalf\(\)](#).

Here is the caller graph for this function:

**13.1.2.65 GetFastInt32RPDF()**

```
int32_t AAX::GetFastInt32RPDF (
    int32_t * iSeed ) [inline]
```

CALL: Calculate pseudo-random 32 bit number based on linear congruential method.

This is required if you want our master bypass functionality in the host to hook up to your bypass parameters.

Parameters

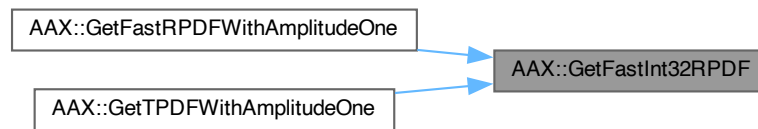
<i>in</i>	<i>iSeed</i>	Seed for random generator
-----------	--------------	---------------------------

Note

This method produces lower quality random numbers (i.e. less random) than plain old GetInt32RPDF, but in many cases it should be plenty good.

Referenced by [GetFastRPDFWithAmplitudeOne\(\)](#), and [GetTPDFWithAmplitudeOne\(\)](#).

Here is the caller graph for this function:

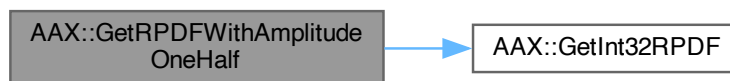


13.1.2.66 GetRPDFWithAmplitudeOneHalf()

```
float AAX::GetRPDFWithAmplitudeOneHalf (
    int32_t * iSeed ) [inline]
```

References [cNormalizeLongToAmplitudeOneHalf](#), and [GetInt32RPDF\(\)](#).

Here is the call graph for this function:

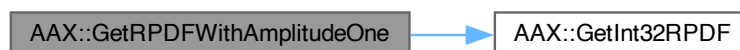


13.1.2.67 GetRPDFWithAmplitudeOne()

```
float AAX::GetRPDFWithAmplitudeOne (
    int32_t * iSeed ) [inline]
```

References [cNormalizeLongToAmplitudeOne](#), and [GetInt32RPDF\(\)](#).

Here is the call graph for this function:

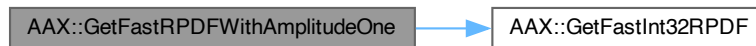


13.1.2.68 GetFastRPDFWithAmplitudeOne()

```
float AAX::GetFastRPDFWithAmplitudeOne (
    int32_t * iSeed ) [inline]
```

References [cNormalizeLongToAmplitudeOne](#), and [GetFastInt32RPDF\(\)](#).

Here is the call graph for this function:

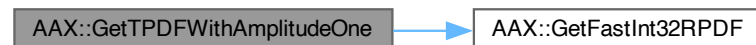


13.1.2.69 GetTPDFWithAmplitudeOne()

```
float AAX::GetTPDFWithAmplitudeOne (
    int32_t * iSeed ) [inline]
```

References [cNormalizeLongToAmplitudeOne](#), and [GetFastInt32RPDF\(\)](#).

Here is the call graph for this function:



13.1.3 Variable Documentation

13.1.3.1 cBigEndian

```
const int AAX::cBigEndian =0
```

13.1.3.2 cLittleEndian

```
const int AAX::cLittleEndian =1
```

13.1.3.3 cPi

```
const double AAX::cPi = 3.1415926535897932384626433832795
```

13.1.3.4 cTwoPi

```
const double AAX::cTwoPi = 6.2831853071795862319959269370884
```

13.1.3.5 cHalfPi

```
const double AAX::cHalfPi = 1.5707963267948965579989817342721
```

Referenced by [SinCosMix\(\)](#).

13.1.3.6 cQuarterPi

```
const double AAX::cQuarterPi = 0.78539816339744827899949086713605
```

13.1.3.7 cRootTwo

```
const double AAX::cRootTwo = 1.4142135623730950488016887242097
```

13.1.3.8 cOneOverRootTwo

```
const double AAX::cOneOverRootTwo = 0.70710678118654752440084436210485
```

13.1.3.9 cPos3dB

```
const double AAX::cPos3dB =1.4142135623730950488016887242097
```

13.1.3.10 cNeg3dB

```
const double AAX::cNeg3dB =0.70710678118654752440084436210485
```

13.1.3.11 cPos6dB

```
const double AAX::cPos6dB =2.0
```

13.1.3.12 cNeg6dB

```
const double AAX::cNeg6dB =0.5
```

13.1.3.13 cNormalizeLongToAmplitudeOneHalf

```
const double AAX::cNormalizeLongToAmplitudeOneHalf = 0.00000000023283064365386962890625
```

Referenced by [GetRPDFWithAmplitudeOneHalf\(\)](#).

13.1.3.14 cNormalizeLongToAmplitudeOne

```
const double AAX::cNormalizeLongToAmplitudeOne = 1.0/double(1<<31)
```

Referenced by [GetFastRPDFWithAmplitudeOne\(\)](#), [GetRPDFWithAmplitudeOne\(\)](#), and [GetTPDFWithAmplitudeOne\(\)](#).

13.1.3.15 cMilli

```
const double AAX::cMilli =0.001
```

13.1.3.16 cMicro

```
const double AAX::cMicro =0.001*0.001
```

13.1.3.17 cNano

```
const double AAX::cNano =0.001*0.001*0.001
```

13.1.3.18 cPico

```
const double AAX::cPico =0.001*0.001*0.001*0.001
```

13.1.3.19 cKilo

```
const double AAX::cKilo =1000.0
```

13.1.3.20 cMega

```
const double AAX::cMega =1000.0*1000.0
```

13.1.3.21 cGiga

```
const double AAX::cGiga =1000.0*1000.0*1000.0
```

13.1.3.22 cDenormalAvoidanceOffset

```
const double AAX::cDenormalAvoidanceOffset =3.0e-34
```

Referenced by [DeDenormal\(\)](#).

13.1.3.23 cFloatDenormalAvoidanceOffset

```
const float AAX::cFloatDenormalAvoidanceOffset = 3.0e-20f
```

Referenced by [DeDenormal\(\)](#).

13.1.3.24 kPowExtent

```
const unsigned int AAX::kPowExtent = 9
```

13.1.3.25 kPowTableSize

```
const unsigned int AAX::kPowTableSize = 1 << kPowExtent
```

13.1.3.26 cSeedDivisor

```
const float AAX::cSeedDivisor = 1/127773.0f
```

Referenced by [GetInt32RPDF\(\)](#).

13.1.3.27 cInitialSeedValue

```
const int32_t AAX::cInitialSeedValue = 0x00F54321
```

13.2 AAX::Exception Namespace Reference

13.2.1 Description

AAX exception classes

All AAX exception classes inherit from [AAX::Exception::Any](#)

Classes

- class [Any](#)
- class [ResultError](#)

13.3 AAX::internal Namespace Reference

Functions

- `template<typename T >`
`std::string ToHexadecimal (T inValue, bool inLeadingZeros=false)`

13.3.1 Function Documentation

13.3.1.1 ToHexadecimal()

```
template<typename T >
std::string AAX::internal::ToHexadecimal (
    T inValue,
    bool inLeadingZeros = false )
```

References [AAX_CONSTEXPR](#).

13.4 AAX_ChunkDataParserDefs Namespace Reference

13.4.1 Description

Constants used by ChunkDataParser class.

Variables

- `const int32_t FLOAT_TYPE = 1`
- `const char FLOAT_STRING_IDENTIFIER [] = "f_"`
- `const int32_t LONG_TYPE = 2`
- `const char LONG_STRING_IDENTIFIER [] = "l_"`
- `const int32_t DOUBLE_TYPE = 3`
- `const char DOUBLE_STRING_IDENTIFIER [] = "d_"`
- `const size_t DOUBLE_TYPE_SIZE = 8`
- `const size_t DOUBLE_TYPE_INCR = 8`
- `const int32_t SHORT_TYPE = 4`
- `const char SHORT_STRING_IDENTIFIER [] = "s_"`
- `const size_t SHORT_TYPE_SIZE = 2`
- `const size_t SHORT_TYPE_INCR = 4`
- `const int32_t STRING_TYPE = 5`
- `const char STRING_STRING_IDENTIFIER [] = "r_"`
- `const size_t MAX_STRINGDATA_LENGTH = 255`
- `const size_t DEFAULT32BIT_TYPE_SIZE = 4`
- `const size_t DEFAULT32BIT_TYPE_INCR = 4`
- `const size_t STRING_IDENTIFIER_SIZE = 2`
- `const int32_t NAME_NOT_FOUND = -1`
- `const size_t MAX_NAME_LENGTH = 255`
- `const int32_t BUILD_DATA_FAILED = -333`
- `const int32_t HEADER_SIZE = 4`
- `const int32_t VERSION_ID_1 = 0x01010101`

13.4.2 Variable Documentation

13.4.2.1 FLOAT_TYPE

```
const int32_t AAX_ChunkDataParserDefs::FLOAT_TYPE = 1
```

13.4.2.2 FLOAT_STRING_IDENTIFIER

```
const char AAX_ChunkDataParserDefs::FLOAT_STRING_IDENTIFIER[] = "f_"
```

13.4.2.3 LONG_TYPE

```
const int32_t AAX_ChunkDataParserDefs::LONG_TYPE = 2
```

13.4.2.4 LONG_STRING_IDENTIFIER

```
const char AAX_ChunkDataParserDefs::LONG_STRING_IDENTIFIER[] = "l_"
```

13.4.2.5 DOUBLE_TYPE

```
const int32_t AAX_ChunkDataParserDefs::DOUBLE_TYPE = 3
```

13.4.2.6 DOUBLE_STRING_IDENTIFIER

```
const char AAX_ChunkDataParserDefs::DOUBLE_STRING_IDENTIFIER[] = "d_"
```

13.4.2.7 DOUBLE_TYPE_SIZE

```
const size_t AAX_ChunkDataParserDefs::DOUBLE_TYPE_SIZE = 8
```

13.4.2.8 DOUBLE_TYPE_INCR

```
const size_t AAX_ChunkDataParserDefs::DOUBLE_TYPE_INCR = 8
```

13.4.2.9 SHORT_TYPE

```
const int32_t AAX_ChunkDataParserDefs::SHORT_TYPE = 4
```

13.4.2.10 SHORT_STRING_IDENTIFIER

```
const char AAX_ChunkDataParserDefs::SHORT_STRING_IDENTIFIER[ ] = "s_"
```

13.4.2.11 SHORT_TYPE_SIZE

```
const size_t AAX_ChunkDataParserDefs::SHORT_TYPE_SIZE = 2
```

13.4.2.12 SHORT_TYPE_INCR

```
const size_t AAX_ChunkDataParserDefs::SHORT_TYPE_INCR = 4
```

13.4.2.13 STRING_TYPE

```
const int32_t AAX_ChunkDataParserDefs::STRING_TYPE = 5
```

13.4.2.14 STRING_STRING_IDENTIFIER

```
const char AAX_ChunkDataParserDefs::STRING_STRING_IDENTIFIER[ ] = "r_"
```

13.4.2.15 MAX_STRINGDATA_LENGTH

```
const size_t AAX_ChunkDataParserDefs::MAX_STRINGDATA_LENGTH = 255
```

13.4.2.16 DEFAULT32BIT_TYPE_SIZE

```
const size_t AAX_ChunkDataParserDefs::DEFAULT32BIT_TYPE_SIZE = 4
```

13.4.2.17 DEFAULT32BIT_TYPE_INCR

```
const size_t AAX_ChunkDataParserDefs::DEFAULT32BIT_TYPE_INCR = 4
```

13.4.2.18 STRING_IDENTIFIER_SIZE

```
const size_t AAX_ChunkDataParserDefs::STRING_IDENTIFIER_SIZE = 2
```

13.4.2.19 NAME_NOT_FOUND

```
const int32_t AAX_ChunkDataParserDefs::NAME_NOT_FOUND = -1
```

13.4.2.20 MAX_NAME_LENGTH

```
const size_t AAX_ChunkDataParserDefs::MAX_NAME_LENGTH = 255
```

13.4.2.21 BUILD_DATA_FAILED

```
const int32_t AAX_ChunkDataParserDefs::BUILD_DATA_FAILED = -333
```

13.4.2.22 HEADER_SIZE

```
const int32_t AAX_ChunkDataParserDefs::HEADER_SIZE = 4
```

13.4.2.23 VERSION_ID_1

```
const int32_t AAX_ChunkDataParserDefs::VERSION_ID_1 = 0x01010101
```

Chapter 14

Class Documentation

14.1 `_acfUID` Struct Reference

Public Attributes

- `uint32_t` [Data1](#)
- `uint16_t` [Data2](#)
- `uint16_t` [Data3](#)
- `uint8_t` [Data4](#) [8]

14.1.1 Member Data Documentation

14.1.1.1 `Data1`

```
uint32_t _acfUID::Data1
```

14.1.1.2 `Data2`

```
uint16_t _acfUID::Data2
```

14.1.1.3 `Data3`

```
uint16_t _acfUID::Data3
```

14.1.1.4 Data4

```
uint8_t _acfUID::Data4[8]
```

The documentation for this struct was generated from the following file:

- [AAX_ACFInterface.doxygen](#)

14.2 AAX_AggregateResult Class Reference

```
#include <AAX_Exception.h>
```

14.2.1 Description

RAII failure count convenience class for use with [AAX_CAPTURE\(\)](#) or [AAX_CAPTURE_MULT\(\)](#)

Pass this object as the first argument in a series of [AAX_CAPTURE\(\)](#) calls to count the number of failures that occur and to re-throw the last error if zero of the attempted calls succeed.

```
// example A: throw if all operations fail
AAX_AggregateResult agg;
AAX_CAPTURE( agg, RegisterThingA(); );
AAX_CAPTURE( agg, RegisterThingB(); );
AAX_CAPTURE( agg, RegisterThingC(); );
```

In this example, when `agg` goes out of scope it checks whether any of A, B, or C succeeded. If none succeeded then the last error that was encountered is raised via an [AAX_CheckedResult::Exception](#). If at least one of the calls succeeded then any failures are swallowed and execution continues as normal. This approach can be useful in cases where you want to run every operation in a group and you only want a failure to be returned if all of the operations failed.

```
// example B: throw if any operation fails
AAX_AggregateResult agg;
AAX_CAPTURE( agg, ImportantOperationW(); );
AAX_CAPTURE( agg, ImportantOperationX(); );
AAX_CAPTURE( agg, ImportantOperationY(); );
AAX_CheckedResult err = agg;
```

In this example, the last error encountered by `agg` is converted to an [AAX_CheckedResult](#). This will result in an [AAX_CheckedResult::Exception](#) even if at least one of the attempted operations succeeded. This approach can be useful in cases where you want all operations in a group to be executed before an error is raised for any failure within the group.

Public Member Functions

- [AAX_AggregateResult](#) ()=default
- [~AAX_AggregateResult](#) ()
- [AAX_AggregateResult & operator= \(AAX_Result inResult\)](#)
Overloaded operator= () for conversion from AAX_Result.
- [operator AAX_Result](#) ()
Implicit conversion to AAX_Result clears the state.
- void [Check](#) () const
- void [Clear](#) ()
- [AAX_Result LastFailure](#) () const
- int [NumFailed](#) () const
- int [NumSucceeded](#) () const
- int [NumAttempted](#) () const

14.2.2 Constructor & Destructor Documentation

14.2.2.1 AAX_AggregateResult()

```
AAX_AggregateResult::AAX_AggregateResult ( ) [default]
```

14.2.2.2 ~AAX_AggregateResult()

```
AAX_AggregateResult::~~AAX_AggregateResult ( ) [inline]
```

References [Check\(\)](#).

Here is the call graph for this function:



14.2.3 Member Function Documentation

14.2.3.1 operator=()

```
AAX_AggregateResult & AAX_AggregateResult::operator= (
    AAX_Result inResult ) [inline]
```

Overloaded `operator=()` for conversion from [AAX_Result](#).

References [AAX_SUCCESS](#).

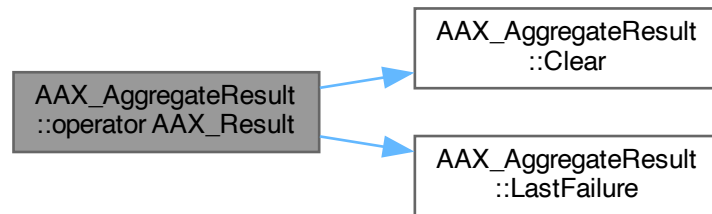
14.2.3.2 operator AAX_Result()

```
AAX_AggregateResult::operator AAX_Result ( ) [inline]
```

Implicit conversion to AAX_Result clears the state.

References [Clear\(\)](#), and [LastFailure\(\)](#).

Here is the call graph for this function:



14.2.3.3 Check()

```
void AAX_AggregateResult::Check ( ) const [inline]
```

Referenced by [~AAX_AggregateResult\(\)](#).

Here is the caller graph for this function:



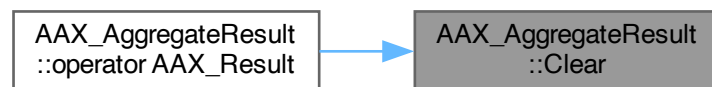
14.2.3.4 Clear()

```
void AAX_AggregateResult::Clear ( ) [inline]
```

References [AAX_SUCCESS](#).

Referenced by [operator AAX_Result\(\)](#).

Here is the caller graph for this function:

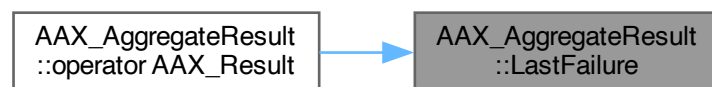


14.2.3.5 LastFailure()

```
AAX_Result AAX_AggregateResult::LastFailure ( ) const [inline]
```

Referenced by [operator AAX_Result\(\)](#).

Here is the caller graph for this function:



14.2.3.6 NumFailed()

```
int AAX_AggregateResult::NumFailed ( ) const [inline]
```

14.2.3.7 NumSucceeded()

```
int AAX_AggregateResult::NumSucceeded ( ) const [inline]
```

14.2.3.8 NumAttempted()

```
int AAX_AggregateResult::NumAttempted ( ) const [inline]
```

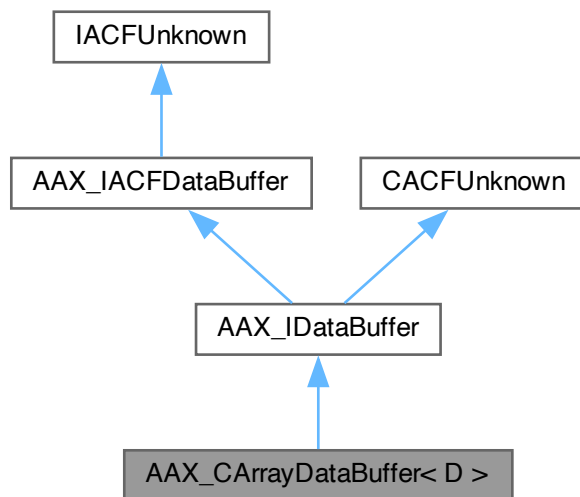
The documentation for this class was generated from the following file:

- [AAX_Exception.h](#)

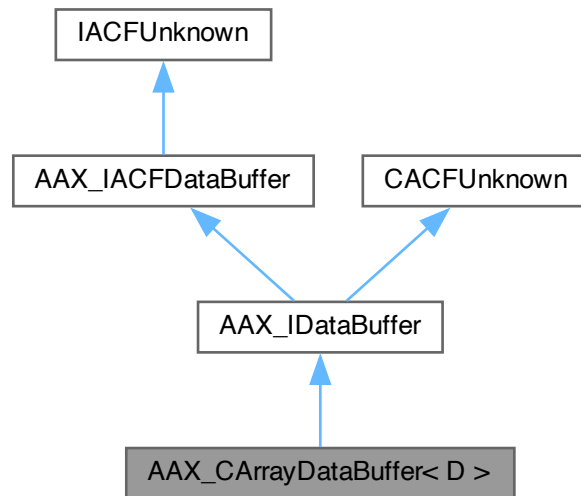
14.3 AAX_CArrayDataBuffer< D > Class Template Reference

```
#include <AAX_CArrayDataBuffer.h>
```

Inheritance diagram for AAX_CArrayDataBuffer< D >:



Collaboration diagram for AAX_CArrayDataBuffer< D >:



14.3.1 Description

```
template<class D>
class AAX_CArrayDataBuffer< D >
```

A convenience class for array data buffers.

The data payload is an array of **D**

Public Member Functions

- [AAX_CArrayDataBuffer](#) ([AAX_CTypeID](#) inType, std::vector< D > const &inData)
- [AAX_CArrayDataBuffer](#) ([AAX_CTypeID](#) inType, std::vector< D > &&inData)
- [AAX_CArrayDataBuffer](#) ([AAX_CArrayDataBuffer](#) const &)=default
- [AAX_CArrayDataBuffer](#) ([AAX_CArrayDataBuffer](#) &&)=default
- [~AAX_CArrayDataBuffer](#) (void) [AAX_OVERRIDE](#)=default
- [AAX_CArrayDataBuffer](#) & operator= ([AAX_CArrayDataBuffer](#) const &other)=default
- [AAX_CArrayDataBuffer](#) & operator= ([AAX_CArrayDataBuffer](#) &&other)=default
- [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const [AAX_OVERRIDE](#)
- [AAX_Result Size](#) (int32_t *oSize) const [AAX_OVERRIDE](#)
- [AAX_Result Data](#) (void const **oBuffer) const [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_IDataBuffer](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid
- [AAX_DELETE](#) ([AAX_IDataBuffer](#) &operator=(const [AAX_IDataBuffer](#) &))
- virtual [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result Size](#) (int32_t *oSize) const =0
- virtual [AAX_Result Data](#) (void const **oBuffer) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Additional Inherited Members

Public Attributes inherited from [AAX_IDataBuffer](#)

- void **ppvObjOut [AAX_OVERRIDE](#)

14.3.2 Constructor & Destructor Documentation

14.3.2.1 [AAX_CArrayDataBuffer\(\)](#) [1/4]

```
template<class D >
AAX_CArrayDataBuffer< D >::AAX_CArrayDataBuffer (
    AAX_CTypeID inType,
    std::vector< D > const & inData ) [inline]
```

14.3.2.2 [AAX_CArrayDataBuffer\(\)](#) [2/4]

```
template<class D >
AAX_CArrayDataBuffer< D >::AAX_CArrayDataBuffer (
    AAX_CTypeID inType,
    std::vector< D > && inData ) [inline]
```

14.3.2.3 [AAX_CArrayDataBuffer\(\)](#) [3/4]

```
template<class D >
AAX_CArrayDataBuffer< D >::AAX_CArrayDataBuffer (
    AAX_CArrayDataBuffer< D > const & ) [default]
```

14.3.2.4 AAX_CArrayDataBuffer() [4/4]

```
template<class D >
AAX_CArrayDataBuffer< D >::AAX_CArrayDataBuffer (
    AAX_CArrayDataBuffer< D > && ) [default]
```

14.3.2.5 ~AAX_CArrayDataBuffer()

```
template<class D >
AAX_CArrayDataBuffer< D >::~~AAX_CArrayDataBuffer (
    void ) [default]
```

14.3.3 Member Function Documentation**14.3.3.1 operator=()** [1/2]

```
template<class D >
AAX_CArrayDataBuffer & AAX_CArrayDataBuffer< D >::operator= (
    AAX_CArrayDataBuffer< D > const & other ) [default]
```

14.3.3.2 operator=() [2/2]

```
template<class D >
AAX_CArrayDataBuffer & AAX_CArrayDataBuffer< D >::operator= (
    AAX_CArrayDataBuffer< D > && other ) [default]
```

14.3.3.3 Type()

```
template<class D >
AAX_Result AAX_CArrayDataBuffer< D >::Type (
    AAX_CTypeID * oType ) const [inline], [virtual]
```

The type of data contained in this buffer

This identifier must be sufficient for a client that knows the type to correctly interpret and use the data.

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

14.3.3.4 Size()

```
template<class D >
AAX_Result AAX_CArrayDataBuffer< D >::Size (
    int32_t * oSize ) const [inline], [virtual]
```

The number of bytes of data in this buffer

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), [AAX_ERROR_SIGNED_INT_OVERFLOW](#), and [AAX_SUCCESS](#).

14.3.3.5 Data()

```
template<class D >
AAX_Result AAX_CArrayDataBuffer< D >::Data (
    void const ** oBuffer ) const [inline], [virtual]
```

The buffer of data

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

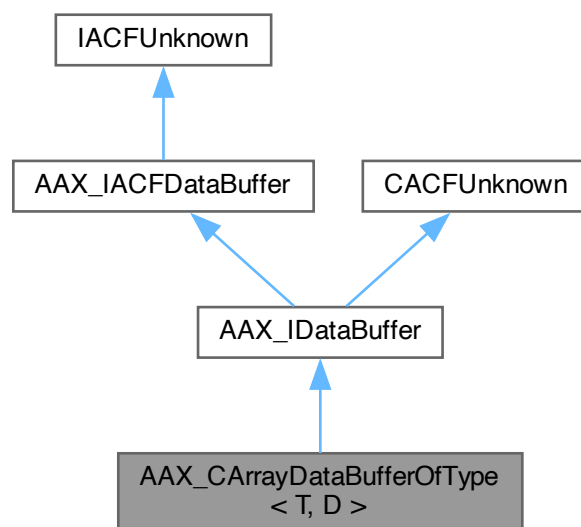
The documentation for this class was generated from the following file:

- [AAX_CArrayDataBuffer.h](#)

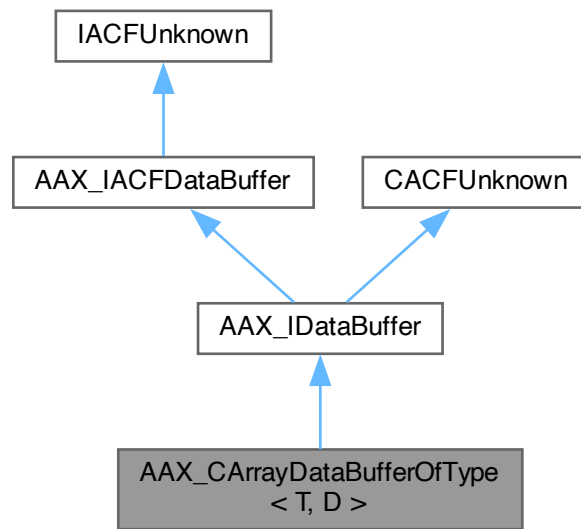
14.4 AAX_CArrayDataBufferOfType< T, D > Class Template Reference

```
#include <AAX_CArrayDataBuffer.h>
```

Inheritance diagram for AAX_CArrayDataBufferOfType< T, D >:



Collaboration diagram for AAX_CArrayDataBufferOfType< T, D >:



14.4.1 Description

```
template<AAX\_CTypeID T, class D>
class AAX_CArrayDataBufferOfType< T, D >
```

A convenience class for array data buffers.

The data payload is an array of `D`

Public Member Functions

- [AAX_CArrayDataBufferOfType](#) (std::vector< D > const &inData)
- [AAX_CArrayDataBufferOfType](#) (std::vector< D > &&inData)
- [AAX_CArrayDataBufferOfType](#) ([AAX_CArrayDataBufferOfType](#) const &)=default
- [AAX_CArrayDataBufferOfType](#) ([AAX_CArrayDataBufferOfType](#) &&)=default
- [~AAX_CArrayDataBufferOfType](#) (void) [AAX_OVERRIDE](#)=default
- [AAX_CArrayDataBufferOfType](#) & operator= ([AAX_CArrayDataBufferOfType](#) const &other)=default
- [AAX_CArrayDataBufferOfType](#) & operator= ([AAX_CArrayDataBufferOfType](#) &&other)=default
- [AAX_Result](#) Type ([AAX_CTypeID](#) *oType) const [AAX_OVERRIDE](#)
- [AAX_Result](#) Size (int32_t *oSize) const [AAX_OVERRIDE](#)
- [AAX_Result](#) Data (void const **oBuffer) const [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_IDataBuffer](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acflID](#) &riid
- [AAX_DELETE](#) ([AAX_IDataBuffer](#) &operator=(const [AAX_IDataBuffer](#) &))
- virtual [AAX_Result](#) Type ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result](#) Size (int32_t *oSize) const =0
- virtual [AAX_Result](#) Data (void const **oBuffer) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Additional Inherited Members

Public Attributes inherited from [AAX_IDataBuffer](#)

- void **ppvObjOut [AAX_OVERRIDE](#)

14.4.2 Constructor & Destructor Documentation

14.4.2.1 [AAX_CArrayDataBufferOfType\(\)](#) [1/4]

```
template<AAX\_CTypeID T, class D >
AAX\_CArrayDataBufferOfType< T, D >::AAX_CArrayDataBufferOfType (
    std::vector< D > const & inData ) [inline], [explicit]
```

14.4.2.2 [AAX_CArrayDataBufferOfType\(\)](#) [2/4]

```
template<AAX\_CTypeID T, class D >
AAX\_CArrayDataBufferOfType< T, D >::AAX_CArrayDataBufferOfType (
    std::vector< D > && inData ) [inline], [explicit]
```

14.4.2.3 [AAX_CArrayDataBufferOfType\(\)](#) [3/4]

```
template<AAX\_CTypeID T, class D >
AAX\_CArrayDataBufferOfType< T, D >::AAX_CArrayDataBufferOfType (
    AAX\_CArrayDataBufferOfType< T, D > const & ) [default]
```


14.4.2.4 AAX_CArrayDataBufferOfType() [4/4]

```
template<AAX_CTypeID T, class D >
AAX_CArrayDataBufferOfType< T, D >::AAX_CArrayDataBufferOfType (
    AAX_CArrayDataBufferOfType< T, D > && ) [default]
```

14.4.2.5 ~AAX_CArrayDataBufferOfType()

```
template<AAX_CTypeID T, class D >
AAX_CArrayDataBufferOfType< T, D >::~~AAX_CArrayDataBufferOfType (
    void ) [default]
```

14.4.3 Member Function Documentation**14.4.3.1 operator=()** [1/2]

```
template<AAX_CTypeID T, class D >
AAX_CArrayDataBufferOfType & AAX_CArrayDataBufferOfType< T, D >::operator= (
    AAX_CArrayDataBufferOfType< T, D > const & other ) [default]
```

14.4.3.2 operator=() [2/2]

```
template<AAX_CTypeID T, class D >
AAX_CArrayDataBufferOfType & AAX_CArrayDataBufferOfType< T, D >::operator= (
    AAX_CArrayDataBufferOfType< T, D > && other ) [default]
```

14.4.3.3 Type()

```
template<AAX_CTypeID T, class D >
AAX_Result AAX_CArrayDataBufferOfType< T, D >::Type (
    AAX_CTypeID * oType ) const [inline], [virtual]
```

The type of data contained in this buffer

This identifier must be sufficient for a client that knows the type to correctly interpret and use the data.

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

14.4.3.4 Size()

```
template<AAX_CTypeID T, class D >
AAX_Result AAX_CArrayDataBufferOfType< T, D >::Size (
    int32_t * oSize ) const [inline], [virtual]
```

The number of bytes of data in this buffer

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), [AAX_ERROR_SIGNED_INT_OVERFLOW](#), and [AAX_SUCCESS](#).

14.4.3.5 Data()

```
template<AAX_CTypeID T, class D >
AAX_Result AAX_CArrayDataBufferOfType< T, D >::Data (
    void const ** oBuffer ) const [inline], [virtual]
```

The buffer of data

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

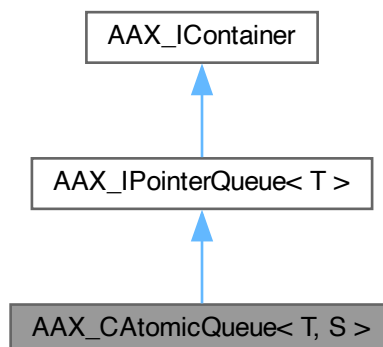
The documentation for this class was generated from the following file:

- [AAX_CArrayDataBuffer.h](#)

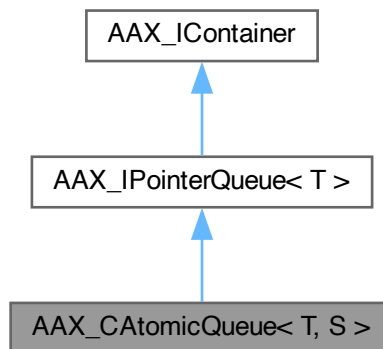
14.5 AAX_CAtomicQueue< T, S > Class Template Reference

```
#include <AAX_CAtomicQueue.h>
```

Inheritance diagram for AAX_CAtomicQueue< T, S >:



Collaboration diagram for AAX_CAtomicQueue< T, S >:



14.5.1 Description

```

template<typename T, size_t S>
class AAX_CAtomicQueue< T, S >

```

Multi-writer, single-reader implementation of [AAX_IPointerQueue](#)

Template parameters:

- **T**: Type of the objects pointed to by this queue
- **S**: Size of the queue's ring buffer. Should be a power of two less than `UINT_32_MAX`

Properties:

- Read operations are non-blocking
- Write operations are synchronized, but very fast
- Supports only one read thread - do not call [Pop\(\)](#) or [Peek\(\)](#) concurrently
- Supports any number of write threads
- Does not support placing `NULL` values onto the queue. [Push](#) will return `eStatus_Unsupported` if a `NULL` value is pushed onto the queue, and the value will be ignored.

Public Types

- typedef [AAX_IPointerQueue< T >::template_type](#) **template_type**
The type used for this template instance.
- typedef [AAX_IPointerQueue< T >::value_type](#) **value_type**
The type of values stored in this queue.

Public Types inherited from [AAX_IPointerQueue< T >](#)

- typedef T [template_type](#)
The type used for this template instance.
- typedef T * [value_type](#)
The type of values stored in this queue.

Public Types inherited from [AAX_IContainer](#)

- enum [EStatus](#) {
[eStatus_Success](#) = 0 ,
[eStatus_Overflow](#) = 1 ,
[eStatus_NotInitialized](#) = 2 ,
[eStatus_Unavailable](#) = 3 ,
[eStatus_Unsupported](#) = 4 }

Public Member Functions

- virtual [~AAX_CAtomicQueue](#) ()
- [AAX_CAtomicQueue](#) ()
- virtual void [Clear](#) ()
- virtual [AAX_IContainer::EStatus Push](#) ([value_type](#) inElem)
- virtual [value_type Pop](#) ()
- virtual [value_type Peek](#) () const

Public Member Functions inherited from [AAX_IPointerQueue< T >](#)

- virtual [~AAX_IPointerQueue](#) ()
- virtual void [Clear](#) ()=0
- virtual [AAX_IContainer::EStatus Push](#) ([value_type](#) inElem)=0
- virtual [value_type Pop](#) ()=0
- virtual [value_type Peek](#) () const =0

Public Member Functions inherited from [AAX_IContainer](#)

- virtual [~AAX_IContainer](#) ()
- virtual void [Clear](#) ()=0

Static Public Attributes

- static const size_t [template_size](#) = S
The size used for this template instance.

14.5.2 Member Typedef Documentation

14.5.2.1 template_type

```
template<typename T , size_t S>
typedef AAX_IPointerQueue<T>::template_type AAX_CAtomicQueue< T, S >::template_type
```

The type used for this template instance.

14.5.2.2 value_type

```
template<typename T , size_t S>
typedef AAX_IPointerQueue<T>::value_type AAX_CAtomicQueue< T, S >::value_type
```

The type of values stored in this queue.

14.5.3 Constructor & Destructor Documentation

14.5.3.1 ~AAX_CAtomicQueue()

```
template<typename T , size_t S>
virtual AAX_CAtomicQueue< T, S >::~~AAX_CAtomicQueue ( ) [inline], [virtual]
```

14.5.3.2 AAX_CAtomicQueue()

```
template<typename T , size_t S>
AAX_CAtomicQueue< T, S >::AAX_CAtomicQueue ( )
```

14.5.4 Member Function Documentation

14.5.4.1 Clear()

```
template<typename T , size_t S>
virtual void AAX_CAtomicQueue< T, S >::Clear ( ) [virtual]
```

Note

This operation is NOT atomic

This does NOT call the destructor for any pointed-to elements; it only clears the pointer values in the queue

Implements [AAX_IPointerQueue< T >](#).

14.5.4.2 Push()

```
template<typename T , size_t S>
virtual AAX_IContainer::EStatus AAX_CAtomicQueue< T, S >::Push (
    value_type inElem ) [virtual]
```

Push an element onto the queue

Call from: Write thread

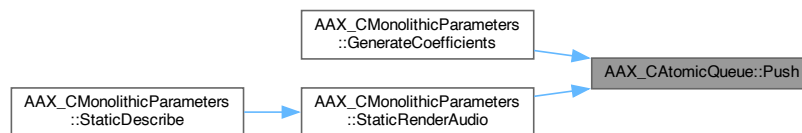
Returns

[AAX_IContainer::EStatus_Success](#) if the push succeeded

Implements [AAX_IPointerQueue< T >](#).

Referenced by [AAX_CMonolithicParameters::GenerateCoefficients\(\)](#), and [AAX_CMonolithicParameters::StaticRenderAudio\(\)](#).

Here is the caller graph for this function:



14.5.4.3 Pop()

```
template<typename T , size_t S>
virtual value_type AAX_CAtomicQueue< T, S >::Pop ( ) [virtual]
```

Pop the front element from the queue

Call from: Read thread

Returns

NULL if no element is available

Implements [AAX_IPointerQueue< T >](#).

14.5.4.4 Peek()

```
template<typename T , size_t S>  
virtual value_type AAX_CAtomicQueue< T, S >::Peek ( ) const [virtual]
```

Get the current top element without popping it off of the queue

Call from: Read thread

Note

This value will change if another thread calls [Pop\(\)](#)

Implements [AAX_IPointerQueue< T >](#).

14.5.5 Member Data Documentation

14.5.5.1 template_size

```
template<typename T , size_t S>  
const size_t AAX_CAtomicQueue< T, S >::template_size = S [static]
```

The size used for this template instance.

The documentation for this class was generated from the following file:

- [AAX_CAtomicQueue.h](#)

14.6 AAX_CAutoreleasePool Class Reference

```
#include <AAX_CAutoreleasePool.h>
```

Public Member Functions

- [AAX_CAutoreleasePool \(\)](#)
- [~AAX_CAutoreleasePool \(\)](#)

14.6.1 Constructor & Destructor Documentation

14.6.1.1 AAX_CAutoreleasePool()

```
AAX_CAutoreleasePool::AAX_CAutoreleasePool ( )
```

14.6.1.2 ~AAX_CAutoreleasePool()

```
AAX_CAutoreleasePool::~~AAX_CAutoreleasePool ( )
```

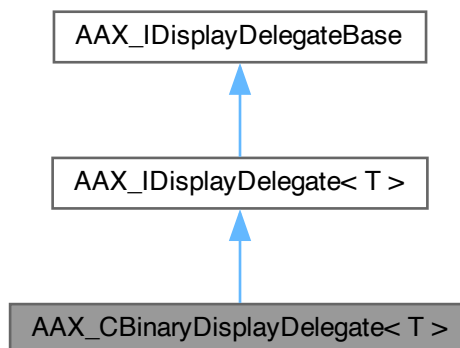
The documentation for this class was generated from the following file:

- [AAX_CAutoreleasePool.h](#)

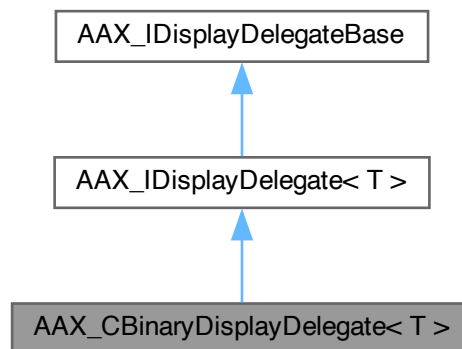
14.7 AAX_CBinaryDisplayDelegate< T > Class Template Reference

```
#include <AAX_CBinaryDisplayDelegate.h>
```

Inheritance diagram for AAX_CBinaryDisplayDelegate< T >:



Collaboration diagram for AAX_CBinaryDisplayDelegate< T >:



14.7.1 Description

```
template<typename T>
class AAX_CBinaryDisplayDelegate< T >
```

A binary display format conforming to [AAX_IDisplayDelegate](#).

This display delegate converts a parameter value to one of two provided strings (e.g. "True" and "False".)

Public Member Functions

- [AAX_CBinaryDisplayDelegate](#) (const char *falseString, const char *trueString)
Constructor.
- [AAX_CBinaryDisplayDelegate](#) (const [AAX_CBinaryDisplayDelegate](#) &other)
- [AAX_IDisplayDelegate< T > * Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- virtual void [AddShortenedStrings](#) (const char *falseString, const char *trueString, int iStrLength)
- virtual [AAX_IDisplayDelegate * Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.7.2 Constructor & Destructor Documentation

14.7.2.1 [AAX_CBinaryDisplayDelegate](#)() [1/2]

```
template<typename T >
AAX_CBinaryDisplayDelegate< T >::AAX_CBinaryDisplayDelegate (
    const char * falseString,
    const char * trueString )
```

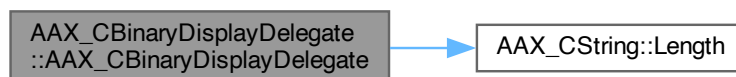
Constructor.

Parameters

in	<i>falseString</i>	The string that will be associated with false parameter values
in	<i>trueString</i>	The string that will be associated with true parameter values

References [AAX_CString::Length\(\)](#).

Here is the call graph for this function:



14.7.2.2 [AAX_CBinaryDisplayDelegate](#)() [2/2]

```
template<typename T >
AAX_CBinaryDisplayDelegate< T >::AAX_CBinaryDisplayDelegate (
    const AAX\_CBinaryDisplayDelegate< T > & other )
```

14.7.3 Member Function Documentation

14.7.3.1 Clone()

```
template<typename T >
AAX_IDisplayDelegate< T > * AAX_CBinaryDisplayDelegate< T >::Clone ( ) const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>* AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.7.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX_CBinaryDisplayDelegate< T >::ValueToString (
    T value,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.7.3.3 ValueToString() [2/2]

```
template<typename T >
bool AAX_CBinaryDisplayDelegate< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.7.3.4 StringToValue()

```
template<typename T >
bool AAX_CBinaryDisplayDelegate< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.7.3.5 AddShortenedStrings()

```
template<typename T >
void AAX_CBinaryDisplayDelegate< T >::AddShortenedStrings (
    const char * falseString,
    const char * trueString,
    int iStrLength ) [virtual]
```

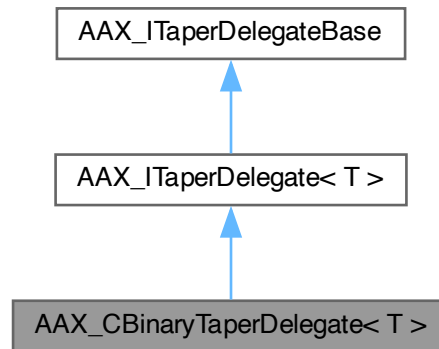
The documentation for this class was generated from the following file:

- [AAX_CBinaryDisplayDelegate.h](#)

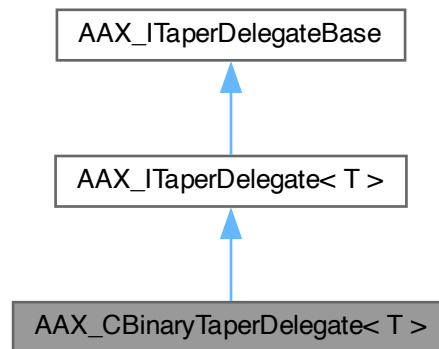
14.8 AAX_CBinaryTaperDelegate< T > Class Template Reference

```
#include <AAX_CBinaryTaperDelegate.h>
```

Inheritance diagram for AAX_CBinaryTaperDelegate< T >:



Collaboration diagram for AAX_CBinaryTaperDelegate< T >:



14.8.1 Description

```
template<typename T>
class AAX_CBinaryTaperDelegate< T >
```

A binary taper conforming to [AAX_ITaperDelegate](#).

This taper maps positive real values to 1 and negative or zero real values to 0. This is the standard taper used on all bool parameters.

When this taper is constructed with a bool template type, its normalized values are automatically typecast to the proper boolean value.

Public Member Functions

- [AAX_CBinaryTaperDelegate](#) ()
Constructs a Binary Taper.
 - [AAX_ITaperDelegate](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the taper delegate.
 - T [GetMaximumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's maximum real value.
 - T [GetMinimumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's minimum real value.
 - T [ConstrainRealValue](#) (T value) const [AAX_OVERRIDE](#)
Applies a constraint to the value and returns the constrained value.
 - T [NormalizedToReal](#) (double normalizedValue) const [AAX_OVERRIDE](#)
Converts a normalized value to a real value.
 - double [RealToNormalized](#) (T realValue) const [AAX_OVERRIDE](#)
Normalizes a real parameter value.
-
- virtual [AAX_ITaperDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the taper delegate.
 - virtual T [GetMaximumValue](#) () const =0
Returns the taper's maximum real value.
 - virtual T [GetMinimumValue](#) () const =0
Returns the taper's minimum real value.
 - virtual T [ConstrainRealValue](#) (T value) const =0
Applies a constraint to the value and returns the constrained value.
 - virtual T [NormalizedToReal](#) (double normalizedValue) const =0
Converts a normalized value to a real value.
 - virtual double [RealToNormalized](#) (T realValue) const =0
Normalizes a real parameter value.

Public Member Functions inherited from [AAX_ITaperDelegateBase](#)

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

14.8.2 Constructor & Destructor Documentation

14.8.2.1 [AAX_CBinaryTaperDelegate](#)()

```
template<typename T >
AAX\_CBinaryTaperDelegate< T >::AAX_CBinaryTaperDelegate
```

Constructs a Binary Taper.

14.8.3 Member Function Documentation

14.8.3.1 Clone()

```
template<typename T >
AAX_ITaperDelegate< T > * AAX_CBinaryTaperDelegate< T >::Clone ( ) const [virtual]
```

Constructs and returns a copy of the taper delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassTaperDelegate<T>* AAX_CSubclassTaperDelegate<T>::Clone() const
{
    return new AAX_CSubclassTaperDelegate(*this);
}
```

Implements [AAX_ITaperDelegate< T >](#).

14.8.3.2 GetMaximumValue()

```
template<typename T >
T AAX_CBinaryTaperDelegate< T >::GetMaximumValue ( ) const [virtual]
```

Returns the taper's maximum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.8.3.3 GetMinimumValue()

```
template<typename T >
T AAX_CBinaryTaperDelegate< T >::GetMinimumValue ( ) const [virtual]
```

Returns the taper's minimum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.8.3.4 ConstrainRealValue()

```
template<typename T >
T AAX_CBinaryTaperDelegate< T >::ConstrainRealValue (
    T value ) const [virtual]
```

Applies a constraint to the value and returns the constrained value.

This method is useful if the taper requires a constraint beyond simple minimum and maximum real value limits.

Note

This is the function that should actually enforces the constraints in [NormalizeToReal\(\)](#) and [RealToNormalized\(\)](#).

Parameters

in	<i>value</i>	The unconstrained value
----	--------------	-------------------------

Implements [AAX_ITaperDelegate< T >](#).

14.8.3.5 NormalizedToReal()

```
template<typename T >
T AAX\_CBinaryTaperDelegate< T >::NormalizedToReal (
    double normalizedValue ) const [virtual]
```

Converts a normalized value to a real value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [RealToNormalized\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>normalizedValue</i>	The normalized value that will be converted
----	------------------------	---

Implements [AAX_ITaperDelegate< T >](#).

14.8.3.6 RealToNormalized()

```
template<typename T >
double AAX\_CBinaryTaperDelegate< T >::RealToNormalized (
    T realValue ) const [virtual]
```

Normalizes a real parameter value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [NormalizedToReal\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>realValue</i>	The real parameter value that will be normalized
----	------------------	--

Implements [AAX_ITaperDelegate< T >](#).

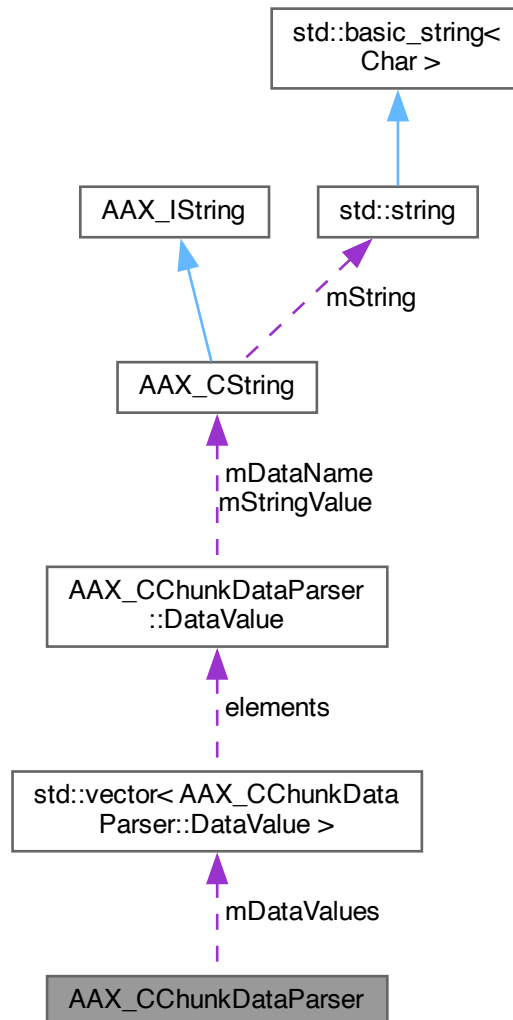
The documentation for this class was generated from the following file:

- [AAX_CBinaryTaperDelegate.h](#)

14.9 AAX_CChunkDataParser Class Reference

```
#include <AAX_CChunkDataParser.h>
```

Collaboration diagram for AAX_CChunkDataParser:



14.9.1 Description

Parser utility for plugin chunks.

Todo Update this documentation for [AAX](#)

This class acts as generic repository for data that is stuffed into or extracted from a SFicPlugInChunk. It has an abstracted Add/Find interface to add or retrieve data values, each uniquely referenced by a c-string. In conjunction with the Effect Layer and the "ControlManager" aspects of the CProcess class, this provides a more transparent & resilient system for performing save-and-restore on settings that won't break so easily from endian issues or from the hard-coded structs that have typically been used to build chunk data.

Format of the Chunk Data

The first 4 bytes of the data are the version number (repeated 4 times to be immune to byte swapping). Data follows next.

Example: "f_bypa %\$#@d_gain #!#@%\$ss_omsi # \$"

type	name	value
float	bypa	%\$#@
double	gain	#!#@%\$ss
int16_t	omsi	# \$

- The first character denotes the data type:

```
'f' = float
'd' = double
'l' = int32
's' = int16
```

- "_" is an empty placekeeper that could be used to addition future information. Currently, it's ignored when a chunk is parsed.
- The string name identifier follows next, and can up to 255 characters int32_t. The Effect Layer builds chunks it always converts the AAX_FourCharCode of the control to a string. So, this will always be 4 characters int32_t. The string is null terminated to indicate the start of the data value.
- The data value follows next, but is possible shifted to aligned word aligned. The size of is determined, of course, by the data type.

Classes

- struct [DataValue](#)

Public Member Functions

- [AAX_CChunkDataParser](#) ()
- virtual [~AAX_CChunkDataParser](#) ()
- void [AddFloat](#) (const char *name, float value)
CALL: Adds some data of type float with name and value to the current chunk.
- void [AddDouble](#) (const char *name, double value)
CALL: See [AddFloat\(\)](#)
- void [AddInt32](#) (const char *name, int32_t value)
CALL: See [AddFloat\(\)](#)
- void [AddInt16](#) (const char *name, int16_t value)
CALL: See [AddFloat\(\)](#)
- void [AddString](#) (const char *name, [AAX_CString](#) value)
- bool [FindFloat](#) (const char *name, float *value)
CALL: Finds some data of type float with name and value in the current chunk.
- bool [FindDouble](#) (const char *name, double *value)
CALL: See [FindFloat\(\)](#)
- bool [FindInt32](#) (const char *name, int32_t *value)
CALL: See [FindFloat\(\)](#)
- bool [FindInt16](#) (const char *name, int16_t *value)
CALL: See [FindFloat\(\)](#)
- bool [FindString](#) (const char *name, [AAX_CString](#) *value)

- bool [ReplaceDouble](#) (const char *name, double value)
- int32_t [GetChunkData](#) ([AAX_SPlugInChunk](#) *chunk)
CALL: Fills passed in chunk with data from current chunk; returns 0 if successful.
- int32_t [GetChunkDataSize](#) ()
CALL: Returns size of current chunk.
- int32_t [GetChunkVersion](#) ()
CALL: Lists fVersion in chunk header for convenience.
- bool [IsEmpty](#) ()
CALL: Returns true if no data is in the chunk.
- void [Clear](#) ()
Resets chunk.

Internal Methods

An Effect Layer plugin can ignore these methods. They are handled by or used internally by the Effect Layer.

- int32_t [mLastFoundIndex](#)
The last index found in the chunk.
- char * [mChunkData](#)
- int32_t [mChunkVersion](#)
Equal to fVersion from the chunk header. Equal to -1 if no chunk is loaded.
- std::vector< [DataValue](#) > [mDataValues](#)
- void [LoadChunk](#) (const [AAX_SPlugInChunk](#) *chunk)
Sets current chunk to data in chunk parameter.
- void [WordAlign](#) (uint32_t &index)
sets index to 4-byte boundary
- void [WordAlign](#) (int32_t &index)
sets index to 4-byte boundary
- int32_t [FindName](#) (const [AAX_CString](#) &Name)
used by public Find methods

14.9.2 Constructor & Destructor Documentation

14.9.2.1 AAX_CChunkDataParser()

```
AAX_CChunkDataParser::AAX_CChunkDataParser ( )
```

14.9.2.2 ~AAX_CChunkDataParser()

```
virtual AAX_CChunkDataParser::~~AAX_CChunkDataParser ( ) [virtual]
```

14.9.3 Member Function Documentation

14.9.3.1 AddFloat()

```
void AAX_CChunkDataParser::AddFloat (
    const char * name,
    float value )
```

CALL: Adds some data of type float with *name* and *value* to the current chunk.

See also

[AddDouble\(\)](#), [AddInt32\(\)](#), and [AddInt16\(\)](#) are the same but with different data types.

14.9.3.2 AddDouble()

```
void AAX_CChunkDataParser::AddDouble (
    const char * name,
    double value )
```

CALL: See [AddFloat\(\)](#)

14.9.3.3 AddInt32()

```
void AAX_CChunkDataParser::AddInt32 (
    const char * name,
    int32_t value )
```

CALL: See [AddFloat\(\)](#)

14.9.3.4 AddInt16()

```
void AAX_CChunkDataParser::AddInt16 (
    const char * name,
    int16_t value )
```

CALL: See [AddFloat\(\)](#)

14.9.3.5 AddString()

```
void AAX_CChunkDataParser::AddString (
    const char * name,
    AAX_CString value )
```

14.9.3.6 FindFloat()

```
bool AAX_CChunkDataParser::FindFloat (
    const char * name,
    float * value )
```

CALL: Finds some data of type float with *name* and *value* in the current chunk.

See also

[FindDouble\(\)](#), [FindInt32\(\)](#), and [FindInt16\(\)](#) are the same but with different data types.

14.9.3.7 FindDouble()

```
bool AAX_CChunkDataParser::FindDouble (
    const char * name,
    double * value )
```

CALL: See [FindFloat\(\)](#)

14.9.3.8 FindInt32()

```
bool AAX_CChunkDataParser::FindInt32 (
    const char * name,
    int32_t * value )
```

CALL: See [FindFloat\(\)](#)

14.9.3.9 FindInt16()

```
bool AAX_CChunkDataParser::FindInt16 (
    const char * name,
    int16_t * value )
```

CALL: See [FindFloat\(\)](#)

14.9.3.10 FindString()

```
bool AAX_CChunkDataParser::FindString (
    const char * name,
    AAX_CString * value )
```

14.9.3.11 ReplaceDouble()

```
bool AAX_CChunkDataParser::ReplaceDouble (
    const char * name,
    double value )
```

14.9.3.12 GetChunkData()

```
int32_t AAX_CChunkDataParser::GetChunkData (
    AAX_SPlugInChunk * chunk )
```

CALL: Fills passed in *chunk* with data from current chunk; returns 0 if successful.

14.9.3.13 GetChunkDataSize()

```
int32_t AAX_CChunkDataParser::GetChunkDataSize ( )
```

CALL: Returns size of current chunk.

14.9.3.14 GetChunkVersion()

```
int32_t AAX_CChunkDataParser::GetChunkVersion ( ) [inline]
```

CALL: Lists fVersion in chunk header for convenience.

References [mChunkVersion](#).

14.9.3.15 IsEmpty()

```
bool AAX_CChunkDataParser::IsEmpty ( )
```

CALL: Returns true if no data is in the chunk.

14.9.3.16 Clear()

```
void AAX_CChunkDataParser::Clear ( )
```

Resets chunk.

14.9.3.17 LoadChunk()

```
void AAX_CChunkDataParser::LoadChunk (
    const AAX_SPlugInChunk * chunk )
```

Sets current chunk to data in *chunk* parameter.

14.9.3.18 WordAlign() [1/2]

```
void AAX_CChunkDataParser::WordAlign (
    uint32_t & index ) [protected]
```

sets *index* to 4-byte boundary

14.9.3.19 WordAlign() [2/2]

```
void AAX_CChunkDataParser::WordAlign (
    int32_t & index ) [protected]
```

sets *index* to 4-byte boundary

14.9.3.20 FindName()

```
int32_t AAX_CChunkDataParser::FindName (
    const AAX_CString & Name ) [protected]
```

used by public Find methods

14.9.4 Member Data Documentation

14.9.4.1 mLastFoundIndex

```
int32_t AAX_CChunkDataParser::mLastFoundIndex [protected]
```

The last index found in the chunk.

Since control values in chunks should tend to stay in order and in sync with the way they're checked with controls within the plug-in, we'll keep track of the value index to speed up searching.

14.9.4.2 mChunkData

```
char* AAX_CChunkDataParser::mChunkData [protected]
```

14.9.4.3 mChunkVersion

```
int32_t AAX_CChunkDataParser::mChunkVersion [protected]
```

Equal to fVersion from the chunk header. Equal to -1 if no chunk is loaded.

Referenced by [GetChunkVersion\(\)](#).

14.9.4.4 mDataValues

```
std::vector<DataValue> AAX_CChunkDataParser::mDataValues
```

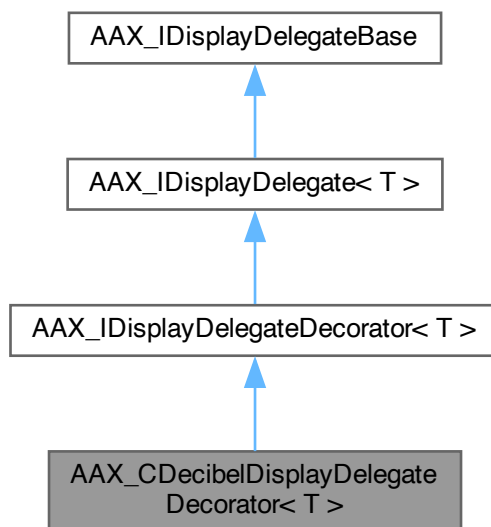
The documentation for this class was generated from the following file:

- [AAX_CChunkDataParser.h](#)

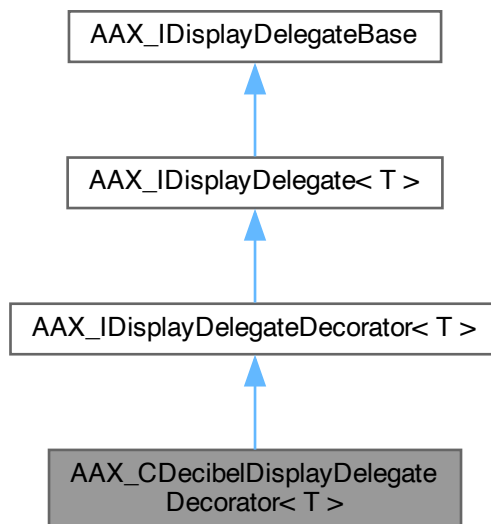
14.10 AAX_CDecibelDisplayDelegateDecorator< T > Class Template Reference

```
#include <AAX_CDecibelDisplayDelegateDecorator.h>
```


Inheritance diagram for AAX_CDecibelDisplayDelegateDecorator< T >:



Collaboration diagram for AAX_CDecibelDisplayDelegateDecorator< T >:



14.10.1 Description

```
template<typename T>
class AAX_CDecibelDisplayDelegateDecorator< T >
```

A percent decorator conforming to [AAX_IDisplayDelegateDecorator](#).

This class is an [AAX_IDisplayDelegateDecorator](#), meaning that it acts as a wrapper for other display delegates or concrete display types. For more information about display delegate decorators in [AAX](#), see [Display delegate decorators](#)

The behavior of this class is to provide conversion to and from dB values. It performs a decibel conversion on the square of the provided value (i.e. 20 log) before passing the value on to a concrete display delegate to get a value string. This class then appends the "dB" suffix to signify that the value was converted. This allows something like a gain value to remain internally linear at all times even though its display is converted to decibels.

The inverse is also supported; this class can convert a decibel-formatted string into its associated real value. The string will first be converted to a number, then that number will have the inverse dB calculation applied to it to retrieve the parameter's real value.

Public Member Functions

- [AAX_CDecibelDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate)
- [AAX_CDecibelDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateDecorator](#)< T >

- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate)
Constructor.
- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegateDecorator](#) &other)
Copy constructor.
- [~AAX_IDisplayDelegateDecorator](#) () [AAX_OVERRIDE](#)
Virtual destructor.
- [AAX_IDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate decorator.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value with a size constraint.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.10.2 Constructor & Destructor Documentation**14.10.2.1 AAX_CDecibelDisplayDelegateDecorator()**

```
template<typename T >
AAX_CDecibelDisplayDelegateDecorator< T >::AAX_CDecibelDisplayDelegateDecorator (
    const AAX\_IDisplayDelegate< T > & displayDelegate )
```

14.10.3 Member Function Documentation**14.10.3.1 Clone()**

```
template<typename T >
AAX_CDecibelDisplayDelegateDecorator< T > * AAX\_CDecibelDisplayDelegateDecorator< T >::Clone
( ) const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>*    AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.10.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX\_CDecibelDisplayDelegateDecorator< T >::ValueToString (
    T value,
    AAX\_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to <i>value</i>

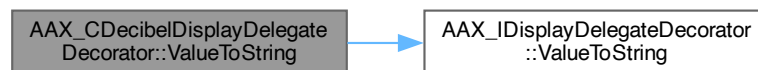
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:



14.10.3.3 ValueToString() [2/2]

```

template<typename T >
bool AAX_CDecibelDisplayDelegateDecorator< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
  
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

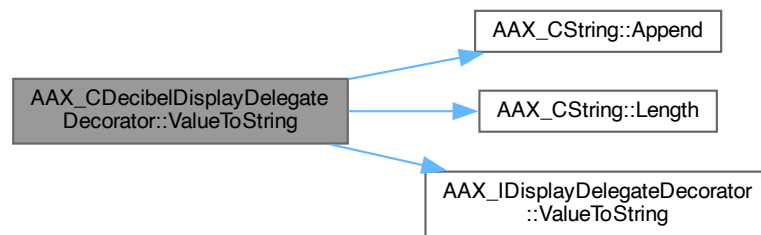
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Append\(\)](#), [AAX_CString::Length\(\)](#), and [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:



14.10.3.4 StringToValue()

```

template<typename T >
bool AAX_CDecibelDisplayDelegateDecorator< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
  
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

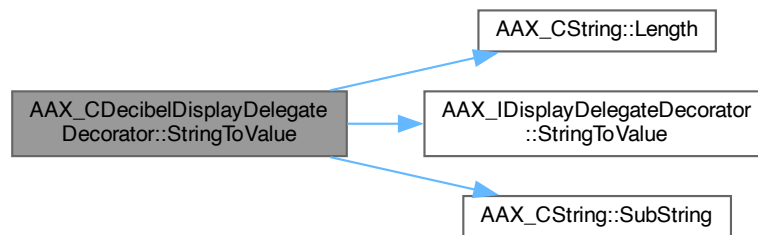
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Length\(\)](#), [AAX_IDisplayDelegateDecorator< T >::StringToValue\(\)](#), and [AAX_CString::SubString\(\)](#).

Here is the call graph for this function:



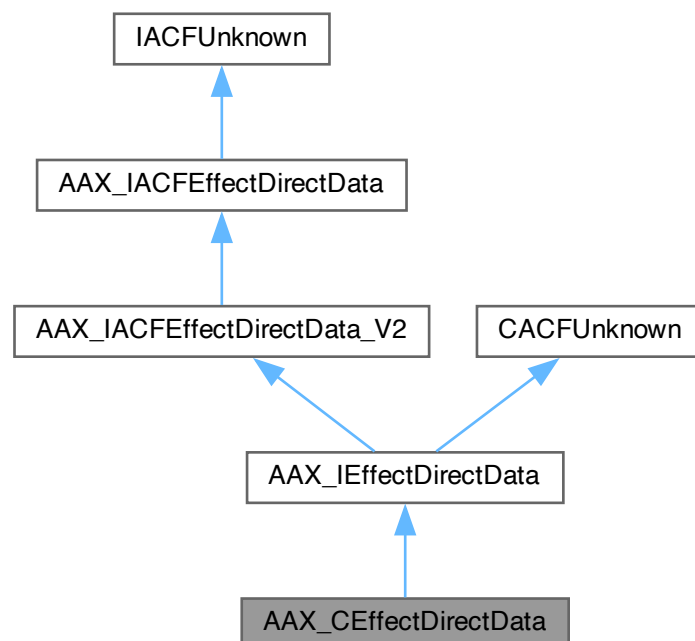
The documentation for this class was generated from the following file:

- [AAX_CDecibelDisplayDelegateDecorator.h](#)

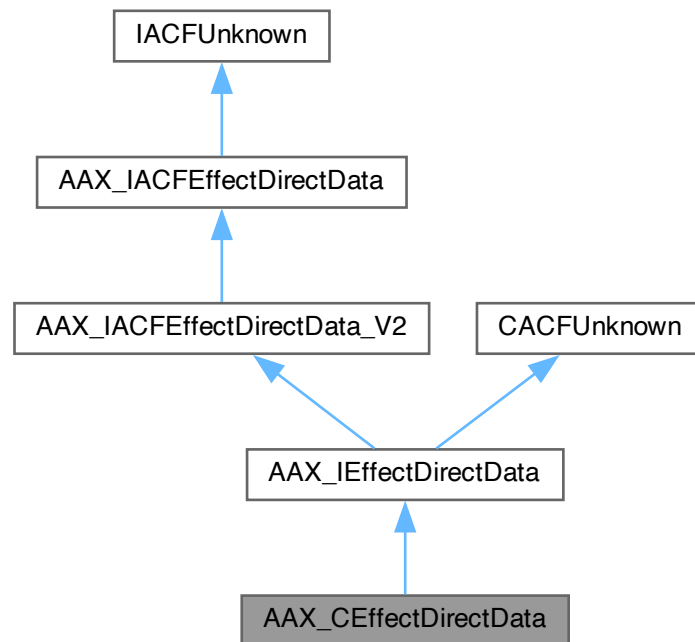
14.11 AAX_CEffectDirectData Class Reference

```
#include <AAX_CEffectDirectData.h>
```

Inheritance diagram for AAX_CEffectDirectData:



Collaboration diagram for AAX_CEffectDirectData:



14.11.1 Description

Default implementation of the [AAX_IEffectDirectData](#) interface.

This class provides a default implementation of the [AAX_IEffectDirectData](#) interface.

Public Member Functions

- [AAX_CEffectDirectData](#) (void)
- virtual [~AAX_CEffectDirectData](#) (void)

Initialization and uninitialization

- [AAX_Result Initialize](#) ([IACFUnknown](#) *iController) [AAX_OVERRIDE](#) [AAX_FINAL](#)
Non-virtual implementation of AAX_IEffectDirectData::Initialize()
- [AAX_Result Uninitialize](#) (void) [AAX_OVERRIDE](#)
Main uninitialization.

Data update callbacks

- [AAX_Result TimerWakeup](#) ([IACFUnknown](#) *iDataAccessInterface) [AAX_OVERRIDE](#)
Non-virtual implementation of AAX_IEffectDirectData::TimerWakeup()

AAX host and plug-in event notification

- [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize) [AAX_OVERRIDE](#)
Notification Hook.

Private member accessors

- [AAX_IController](#) * [Controller](#) (void)
Returns a pointer to the plug-in's controller interface.
- [AAX_IEffectParameters](#) * [EffectParameters](#) (void)
Returns a pointer to the plug-in's data model interface.

Public Member Functions inherited from [AAX_IEffectDirectData](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid
- [AAX_DELETE](#) ([AAX_IEffectDirectData](#) &operator=(const [AAX_IEffectDirectData](#) &))

AAX host and plug-in event notification

Initialization and uninitialization

Safe data update callbacks

These callbacks provide a safe context from which to directly access the algorithm's private data blocks. Each callback is called regularly with roughly the schedule of its corresponding [AAX_IEffectParameters](#) counterpart.

Note

Do not attempt to directly access the algorithm's data from outside these callbacks. Instead, use the packet system to route data to the algorithm using the AAX host's buffered data transfer facilities.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

[AAX_CEffectDirectData](#) virtual interface

- virtual [AAX_Result Initialize_PrivateDataAccess](#) ()
Initialization routine for classes that inherit from [AAX_CEffectDirectData](#). This method is called by the default [Initialize\(\)](#) implementation after all internal members have been initialized, and provides a safe location in which to perform any additional initialization tasks.
- virtual [AAX_Result TimerWakeup_PrivateDataAccess](#) ([AAX_IPrivateDataAccess](#) *iPrivateDataAccess)
Callback provided with an [AAX_IPrivateDataAccess](#). Override this method to access the algorithm's private data using the [AAX_IPrivateDataAccess](#) interface.

Additional Inherited Members

Public Attributes inherited from [AAX_IEffectDirectData](#)

- void **ppvObjOut [override](#)

14.11.2 Constructor & Destructor Documentation

14.11.2.1 AAX_CEffectDirectData()

```
AAX_CEffectDirectData::AAX_CEffectDirectData (
    void )
```

14.11.2.2 ~AAX_CEffectDirectData()

```
virtual AAX_CEffectDirectData::~~AAX_CEffectDirectData (
    void ) [virtual]
```

14.11.3 Member Function Documentation

14.11.3.1 Initialize()

```
AAX_Result AAX_CEffectDirectData::Initialize (
    IACFUnknown * iController ) [virtual]
```

Non-virtual implementation of [AAX_IEfectDirectData::Initialize\(\)](#)

This implementation initializes all private [AAX_CEffectDirectData](#) members and calls [Initialize_PrivateDataAccess\(\)](#). For custom initialization, inherited classes should override [Initialize_PrivateDataAccess\(\)](#).

Parameters

in	<i>iController</i>	Unknown pointer that resolves to an AAX_IController .
----	--------------------	---

Implements [AAX_IACFEfectDirectData](#).

14.11.3.2 Uninitialize()

```
AAX_Result AAX_CEffectDirectData::Uninitialize (
    void ) [virtual]
```

Main uninitialization.

Called when the interface is destroyed.

Implements [AAX_IACFEffectDirectData](#).

14.11.3.3 TimerWakeup()

```
AAX_Result AAX_CEffectDirectData::TimerWakeup (
    IACFUnknown * iDataAccessInterface ) [virtual]
```

Non-virtual implementation of [AAX_IEfectDirectData::TimerWakeup\(\)](#)

This implementation interprets the [IACFUnknown](#) and forwards the resulting [AAX_IPrivateDataAccess](#) to [TimerWakeup_PrivateDataAccess\(\)](#)

Parameters

in	<i>iDataAccessInterface</i>	Unknown pointer that resolves to an AAX_IPrivateDataAccess . This interface is only valid for the duration of this method's execution and is discarded when the method returns.
----	-----------------------------	---

Implements [AAX_IACFEffectDirectData](#).

14.11.3.4 NotificationReceived()

```
AAX_Result AAX_CEffectDirectData::NotificationReceived (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- some notifications are sent only to the plug-in GUI or plug-in data model while other notifications are sent only to the EffectDirectData. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of inNotificationData, in bytes

Implements [AAX_IACFEffDirectData_V2](#).

14.11.3.5 Controller()

```
AAX_IController * AAX_CEffectDirectData::Controller (
    void )
```

Returns a pointer to the plug-in's controller interface.

Todo Change to GetController to match other AAX_CEffect modules

14.11.3.6 EffectParameters()

```
AAX_IEffectParameters * AAX_CEffectDirectData::EffectParameters (
    void )
```

Returns a pointer to the plug-in's data model interface.

Todo Change to GetController to match other AAX_CEffect modules

14.11.3.7 Initialize_PrivateDataAccess()

```
virtual AAX_Result AAX_CEffectDirectData::Initialize_PrivateDataAccess ( ) [protected], [virtual]
```

Initialization routine for classes that inherit from [AAX_CEffectDirectData](#). This method is called by the default [Initialize\(\)](#) implementation after all internal members have been initialized, and provides a safe location in which to perform any additional initialization tasks.

14.11.3.8 TimerWakeup_PrivateDataAccess()

```
virtual AAX_Result AAX_CEffectDirectData::TimerWakeup_PrivateDataAccess (
    AAX_IPrivateDataAccess * iPrivateDataAccess ) [protected], [virtual]
```

Callback provided with an [AAX_IPrivateDataAccess](#). Override this method to access the algorithm's private data using the [AAX_IPrivateDataAccess](#) interface.

Parameters

in	<i>iPrivateDataAccess</i>	Pointer to an AAX_IPrivateDataAccess interface. This interface is only valid for the duration of this method.
----	---------------------------	---

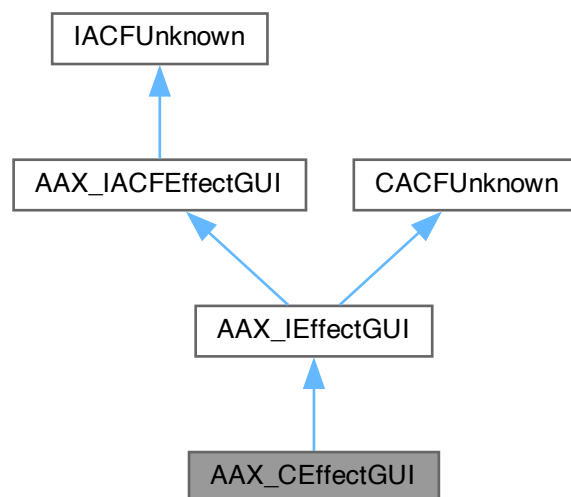
The documentation for this class was generated from the following file:

- [AAX_CEffectDirectData.h](#)

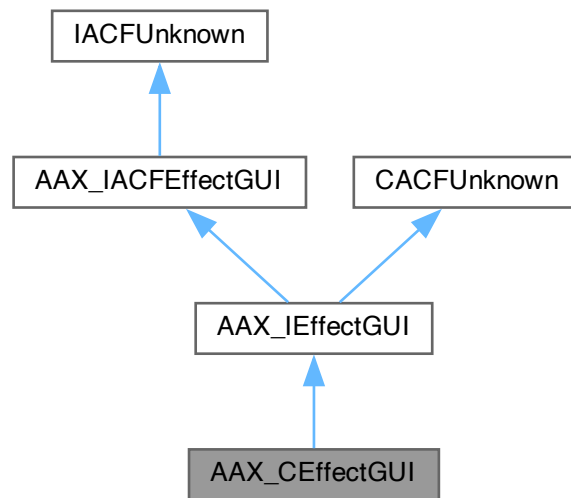
14.12 AAX_CEffectGUI Class Reference

```
#include <AAX_CEffectGUI.h>
```

Inheritance diagram for AAX_CEffectGUI:



Collaboration diagram for AAX_CEffectGUI:



14.12.1 Description

Default implementation of the [AAX_IEffectGUI](#) interface.

This class provides a default implementation of the [AAX_IEffectGUI](#) interface.

Legacy Porting Notes The default implementations in this class are mostly derived from their equivalent implementations in CProcess and CEffectProcess. For additional CProcess-derived implementations, see [AAX_CEffectParameters](#).

Note

See [AAX_IACFEffectGUI](#) for further information.

Public Member Functions

- [AAX_CEffectGUI](#) (void)
- [~AAX_CEffectGUI](#) (void) [AAX_OVERRIDE](#)

Initialization and uninitialization

- [AAX_Result Initialize](#) ([IACFUnknown](#) *iController) [AAX_OVERRIDE](#)
Main GUI initialization.
- [AAX_Result Uninitialize](#) (void) [AAX_OVERRIDE](#)
Main GUI uninitialization.

AAX host and plug-in event notification

- [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize) [AAX_OVERRIDE](#)
Notification Hook.

View accessors

- [AAX_Result SetViewContainer](#) ([IACFUnknown](#) *iViewContainer) [AAX_OVERRIDE](#)
Provides a handle to the main plug-in window.
- [AAX_Result GetViewSize](#) ([AAX_Point](#) *) const [AAX_OVERRIDE](#)
Retrieves the size of the plug-in window.

GUI update methods

- [AAX_Result Draw](#) ([AAX_Rect](#) *) [AAX_OVERRIDE](#)
DEPRECATED, Not called from host any longer. Your chosen graphics framework should be directly handling draw events from the OS.
- [AAX_Result TimerWakeup](#) (void) [AAX_OVERRIDE](#)
Periodic wakeup callback for idle-time operations.
- [AAX_Result ParameterUpdated](#) ([AAX_CParamID](#) paramID) [AAX_OVERRIDE](#)
Notifies the GUI that a parameter value has changed.

Host interface methods

Miscellaneous methods to provide host-specific functionality

- [AAX_Result GetCustomLabel](#) ([AAX_EPlugInStrings](#) iSelector, [AAX_IString](#) *oString) const [AAX_OVERRIDE](#)
Called by host application to retrieve a custom plug-in string.
- [AAX_Result SetControlHighlightInfo](#) ([AAX_CParamID](#), [AAX_CBoolean](#), [AAX_EHighlightColor](#)) [AAX_OVERRIDE](#)
Called by host application. Indicates that a control widget should be updated with a highlight color.

Public Member Functions inherited from [AAX_IEffectGUI](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfIID](#) &riid
- [AAX_DELETE](#) ([AAX_IEffectGUI](#) &operator=(const [AAX_IEffectGUI](#) &))

Initialization and uninitialization

AAX host and plug-in event notification

View accessors

GUI update methods

Host interface methods

Miscellaneous methods to provide host-specific functionality

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Protected Member Functions

AAX_CEffectGUI pure virtual interface

The implementations of these methods will be specific to the particular GUI framework that is being incorporated with [AAX_CEffectGUI](#). Classes that inherit from [AAX_CEffectGUI](#) must override these methods with their own framework-specific implementations.

- virtual void [CreateViewContents](#) (void)=0
Creates any required top-level GUI components.
- virtual void [CreateViewContainer](#) (void)=0
Initializes the plug-in window and creates the main GUI view or frame.
- virtual void [DeleteViewContainer](#) (void)=0
Uninitializes the plug-in window and deletes the main GUI view or frame.

Helper methods

- virtual void [UpdateAllParameters](#) (void)
Requests an update to the GUI for every parameter view.

Private member accessors

- [AAX_IController](#) * [GetController](#) (void)
Retrieves a reference to the plug-in's controller interface.
- const [AAX_IController](#) * [GetController](#) (void) const
- [AAX_IEffectParameters](#) * [GetEffectParameters](#) (void)
Retrieves a reference to the plug-in's data model interface.
- const [AAX_IEffectParameters](#) * [GetEffectParameters](#) (void) const
- [AAX_IViewContainer](#) * [GetViewContainer](#) (void)
Retrieves a reference to the plug-in's view container interface.
- const [AAX_IViewContainer](#) * [GetViewContainer](#) (void) const
- [AAX_ITransport](#) * [Transport](#) ()
Retrieves a reference to the plug-in's Transport interface.
- const [AAX_ITransport](#) * [Transport](#) () const
- [AAX_EViewContainer_Type](#) [GetViewContainerType](#) ()
Retrieves the Container and it's type.
- void * [GetViewContainerPtr](#) ()

Additional Inherited Members

Public Attributes inherited from [AAX_IEffectGUI](#)

- void **ppvObjOut [override](#)

14.12.2 Constructor & Destructor Documentation

14.12.2.1 AAX_CEffectGUI()

```
AAX_CEffectGUI::AAX_CEffectGUI (
    void )
```

14.12.2.2 ~AAX_CEffectGUI()

```
AAX_CEffectGUI::~~AAX_CEffectGUI (
    void )
```

14.12.3 Member Function Documentation

14.12.3.1 Initialize()

```
AAX_Result AAX_CEffectGUI::Initialize (
    IACFUnknown * iController ) [virtual]
```

Main GUI initialization.

Called when the GUI is created

Parameters

in	<i>iController</i>	A versioned reference that resolves to an AAX_IController interface
----	--------------------	---

Implements [AAX_IACFEfectGUI](#).

14.12.3.2 Uninitialize()

```
AAX_Result AAX_CEffectGUI::Uninitialize (
    void ) [virtual]
```

Main GUI uninitialization.

Called when the GUI is destroyed. Frees the GUI.

Implements [AAX_IACFEfectGUI](#).

14.12.3.3 NotificationReceived()

```
AAX_Result AAX_CEffectGUI::NotificationReceived (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- some notifications are sent only to the plug-in GUI while other notifications are sent only to the plug-in data model. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

A plug-in may also dispatch custom notifications using [AAX_IController::SendNotification\(\)](#). Custom notifications will be posted back to the plug-in's other objects which support a [NotificationReceived\(\)](#) method (e.g. the data model).

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Note

The default implementation doesn't do anything at this point, but it is probably still a good idea to call into the base class [AAX_CEffectGUI::NotificationReceived\(\)](#) function in case we want to implement some default behaviors in the future.

Implements [AAX_IACFEffctGUI](#).

14.12.3.4 SetViewContainer()

```
AAX_Result AAX_CEffectGUI::SetViewContainer (
    IACFUnknown * iViewContainer ) [virtual]
```

Provides a handle to the main plug-in window.

Parameters

in	<i>iViewController</i>	An AAX_IViewController providing a native handle to the plug-in's window
----	------------------------	--

Implements [AAX_IACFEEffectGUI](#).

14.12.3.5 GetViewSize()

```
AAX_Result AAX_CEffectGUI::GetViewSize (
    AAX_Point * oViewSize ) const [inline], [virtual]
```

Retrieves the size of the plug-in window.

See also

[AAX_IViewController::SetViewSize\(\)](#)

Parameters

out	<i>oViewSize</i>	The size of the plug-in window as a point (width, height)
-----	------------------	---

Implements [AAX_IACFEEffectGUI](#).

References [AAX_SUCCESS](#).

14.12.3.6 Draw()

```
AAX_Result AAX_CEffectGUI::Draw (
    AAX_Rect * iDrawRect ) [inline], [virtual]
```

DEPRECATED, Not called from host any longer. Your chosen graphics framework should be directly handling draw events from the OS.

Implements [AAX_IACFEEffectGUI](#).

References [AAX_SUCCESS](#).

14.12.3.7 TimerWakeup()

```
AAX_Result AAX_CEffectGUI::TimerWakeup (
    void ) [inline], [virtual]
```

Periodic wakeup callback for idle-time operations.

GUI animation events such as meter updates should be triggered from this method.

This method is called from the host's main thread. In general, it should be driven at approximately one call per 30 ms. However, the wakeup is not guaranteed to be called at any regular interval - for example, it could be held off by a high real-time processing load - and there is no host contract regarding maximum latency between wakeup calls.

This wakeup runs continuously and cannot be armed/disarmed by the plug-in.

Implements [AAX_IACFEffectGUI](#).

References [AAX_SUCCESS](#).

14.12.3.8 ParameterUpdated()

```
AAX_Result AAX_CEffectGUI::ParameterUpdated (
    AAX_CParamID inParamID ) [virtual]
```

Notifies the GUI that a parameter value has changed.

This method is called by the host whenever a parameter value has been modified

This method may be called on a non-main thread

Implements [AAX_IACFEffectGUI](#).

14.12.3.9 GetCustomLabel()

```
AAX_Result AAX_CEffectGUI::GetCustomLabel (
    AAX_EPlugInStrings iSelector,
    AAX_IString * oString ) const [virtual]
```

Called by host application to retrieve a custom plug-in string.

If no string is provided then the host's default will be used.

Parameters

in	<i>iSelector</i>	The requested strong. One of AAX_EPlugInStrings
out	<i>oString</i>	The plug-in's custom value for the requested string

Implements [AAX_IACFEffectGUI](#).

14.12.3.10 SetControlHighlightInfo()

```
AAX_Result AAX_CEffectGUI::SetControlHighlightInfo (
    AAX_CParamID iParameterID,
    AAX_CBoolean iIsHighlighted,
    AAX_EHighlightColor iColor ) [inline], [virtual]
```

Called by host application. Indicates that a control widget should be updated with a highlight color.

Todo Document this method

Legacy Porting Notes This method was re-named from `SetControlHighliteInfo()`, its name in the legacy plug-in SDK.

Parameters

in	<i>iParameterID</i>	ID of parameter whose widget(s) must be highlighted
in	<i>iIsHighlighted</i>	True if turning highlight on, false if turning it off
in	<i>iColor</i>	Desired highlight color. One of AAX_EHighlightColor

Implements [AAX_IACFEEffectGUI](#).

References [AAX_SUCCESS](#).

14.12.3.11 CreateViewContents()

```
virtual void AAX_CEffectGUI::CreateViewContents (
    void ) [protected], [pure virtual]
```

Creates any required top-level GUI components.

This method is called by default from [AAX_CEffectGUI::Initialize\(\)](#)

14.12.3.12 CreateViewContainer()

```
virtual void AAX_CEffectGUI::CreateViewContainer (
    void ) [protected], [pure virtual]
```

Initializes the plug-in window and creates the main GUI view or frame.

This method is called by default from [AAX_CEffectGUI::SetViewContainer\(\)](#) when a valid window is present

14.12.3.13 DeleteViewContainer()

```
virtual void AAX_CEffectGUI::DeleteViewContainer (
    void ) [protected], [pure virtual]
```

Uninitializes the plug-in window and deletes the main GUI view or frame.

This method is called by default from [AAX_CEffectGUI::SetViewContainer\(\)](#) when no valid window is present. It may also be appropriate for inheriting classes to call this method from their destructors, depending on their own internal implementation.

14.12.3.14 UpdateAllParameters()

```
virtual void AAX_CEffectGUI::UpdateAllParameters (
    void ) [protected], [virtual]
```

Requests an update to the GUI for every parameter view.

By default, calls [AAX_CEffectGUI::ParameterUpdated\(\)](#) on every registered parameter.

By default, called from [AAX_CEffectGUI::SetViewContainer\(\)](#) after a new view container has been created.

Todo Rename to `UpdateAllParameterViews()` or another name that does not lead to confusion regarding what exactly this method should be doing.

14.12.3.15 GetController() [1/2]

```
AAX_IController * AAX_CEffectGUI::GetController (
    void )
```

Retrieves a reference to the plug-in's controller interface.

14.12.3.16 GetController() [2/2]

```
const AAX_IController * AAX_CEffectGUI::GetController (
    void ) const
```

14.12.3.17 GetEffectParameters() [1/2]

```
AAX_IEffectParameters * AAX_CEffectGUI::GetEffectParameters (
    void )
```

Retrieves a reference to the plug-in's data model interface.

14.12.3.18 GetEffectParameters() [2/2]

```
const AAX_IEffectParameters * AAX_CEffectGUI::GetEffectParameters (
    void ) const
```

14.12.3.19 GetViewContainer() [1/2]

```
AAX_IViewContainer * AAX_CEffectGUI::GetViewContainer (
    void )
```

Retrieves a reference to the plug-in's view container interface.

14.12.3.20 GetViewContainer() [2/2]

```
const AAX_IViewContainer * AAX_CEffectGUI::GetViewContainer (
    void ) const
```

14.12.3.21 Transport() [1/2]

```
AAX_ITransport * AAX_CEffectGUI::Transport ( )
```

Retrieves a reference to the plug-in's Transport interface.

14.12.3.22 Transport() [2/2]

```
const AAX_ITransport * AAX_CEffectGUI::Transport ( ) const
```

14.12.3.23 GetViewContainerType()

```
AAX_EViewContainer_Type AAX_CEffectGUI::GetViewContainerType ( )
```

Retrieves the Container and it's type.

14.12.3.24 GetViewContainerPtr()

```
void * AAX_CEffectGUI::GetViewContainerPtr ( )
```

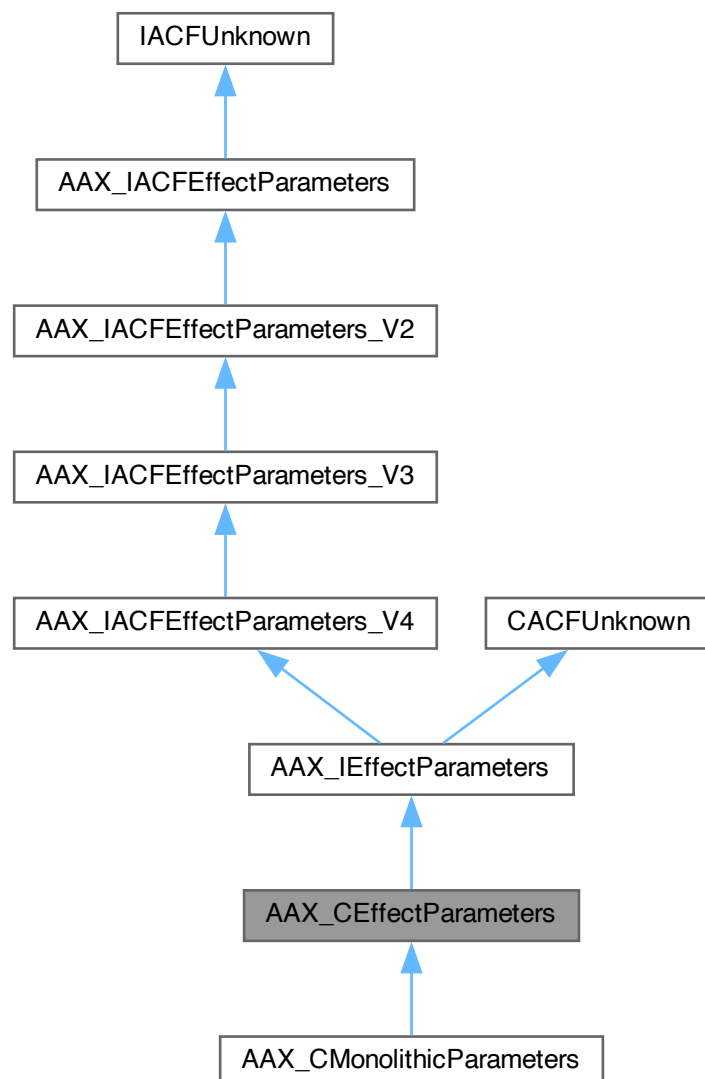
The documentation for this class was generated from the following file:

- [AAX_CEffectGUI.h](#)

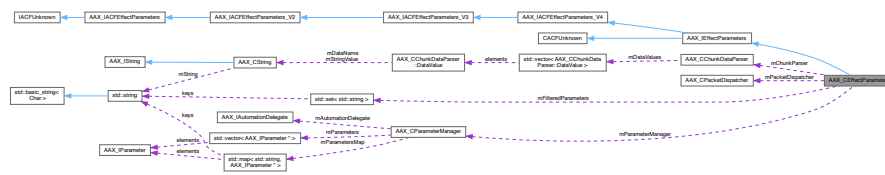
14.13 AAX_CEffectParameters Class Reference

```
#include <AAX_CEffectParameters.h>
```

Inheritance diagram for AAX_CEffectParameters:



Collaboration diagram for AAX_CEffectParameters:



14.13.1 Description

Default implementation of the [AAX_IEffectParameters](#) interface.

This class provides a default implementation of the [AAX_IEffectParameters](#) interface. In nearly all cases, your plug-in's data model should inherit from this class and override only those functions that you wish to explicitly customize.

Legacy Porting Notes The default implementations in this class are mostly derived from their equivalent implementations in CProcess and CEffectProcess. For additional CProcess-derived implementations, see [AAX_CEffectGUI](#).

14.13.2 Related classes

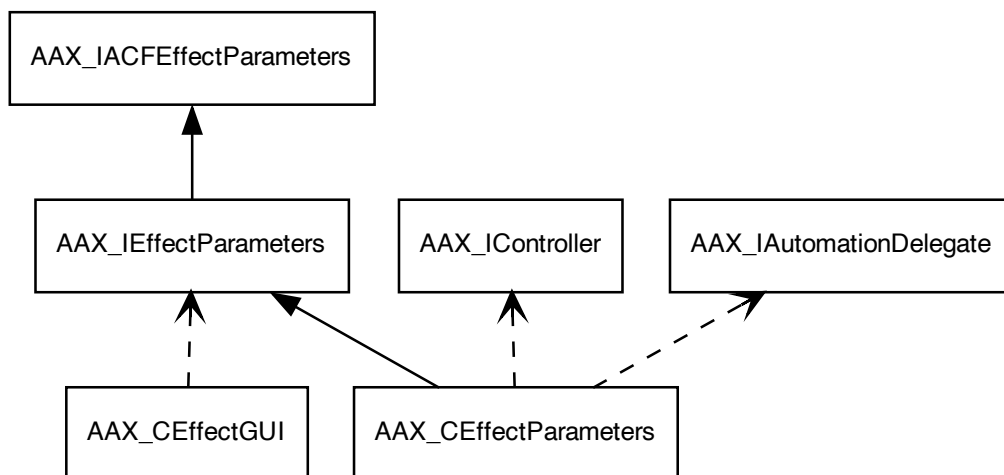


Figure 14.1 Classes related to [AAX_IEffectParameters](#) by inheritance or composition

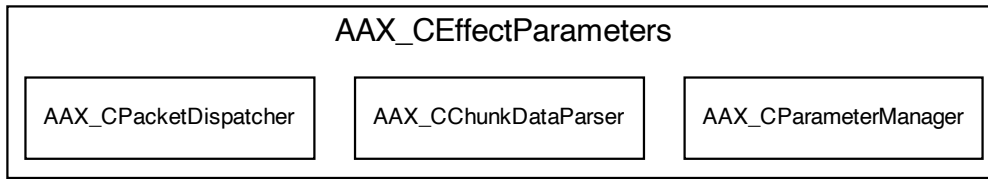


Figure 14.2 Classes owned as member objects of AAX_CEffectParameters

Public Member Functions

- [AAX_CEffectParameters](#) (void)
- [~AAX_CEffectParameters](#) (void) [AAX_OVERRIDE](#)
- [AAX_CEffectParameters](#) & [operator=](#) (const [AAX_CEffectParameters](#) &other)

Initialization and uninitialization

- [AAX_Result Initialize](#) (IACFUnknown *iController) [AAX_OVERRIDE](#)
Main data model initialization. Called when plug-in instance is first instantiated.
- [AAX_Result Uninitialize](#) (void) [AAX_OVERRIDE](#)
Main data model uninitialization.

AAX host and plug-in event notification

- [AAX_Result NotificationReceived](#) (AAX_CTypeID inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize) [AAX_OVERRIDE](#)
Notification Hook.

Parameter information

These methods are used by the AAX host to retrieve information about the plug-in's data model. For information about adding parameters to the plug-in and otherwise modifying the plug-in's data model, see [AAX_CParameterManager](#). For information about parameters, see [AAX_IParameter](#).

- [AAX_Result GetNumberOfParameters](#) (int32_t *oNumControls) const [AAX_OVERRIDE](#)
CALL: Retrieves the total number of plug-in parameters.
- [AAX_Result GetMasterBypassParameter](#) (AAX_IString *oIDString) const [AAX_OVERRIDE](#)
CALL: Retrieves the ID of the plug-in's Master Bypass parameter.
- [AAX_Result GetParameterIsAutomatable](#) (AAX_CParamID iParameterID, AAX_CBoolean *oAutomatable) const [AAX_OVERRIDE](#)
CALL: Retrieves information about a parameter's automatable status.
- [AAX_Result GetParameterNumberOfSteps](#) (AAX_CParamID iParameterID, int32_t *oNumSteps) const [AAX_OVERRIDE](#)
CALL: Retrieves the number of discrete steps for a parameter.
- [AAX_Result GetParameterName](#) (AAX_CParamID iParameterID, AAX_IString *oName) const [AAX_OVERRIDE](#)
CALL: Retrieves the full name for a parameter.
- [AAX_Result GetParameterNameOfLength](#) (AAX_CParamID iParameterID, AAX_IString *oName, int32_t iNameLength) const [AAX_OVERRIDE](#)
CALL: Retrieves an abbreviated name for a parameter.

- [AAX_Result GetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double *oValue) const [AAX_OVERRIDE](#)
CALL: Retrieves default value of a parameter.
- [AAX_Result SetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue) [AAX_OVERRIDE](#)
CALL: Sets the default value of a parameter.
- [AAX_Result GetParameterType](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterType](#) *oParameterType) const [AAX_OVERRIDE](#)
CALL: Retrieves the type of a parameter.
- [AAX_Result GetParameterOrientation](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterOrientation](#) *oParameterOrientation) const [AAX_OVERRIDE](#)
CALL: Retrieves the orientation that should be applied to a parameter's controls.
- [AAX_Result GetParameter](#) ([AAX_CParamID](#) iParameterID, [AAX_IParameter](#) **oParameter) [AAX_OVERRIDE](#)
CALL: Retrieves an arbitrary setting within a parameter.
- [AAX_Result GetParameterIndex](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) *oControllIndex) const [AAX_OVERRIDE](#)
CALL: Retrieves the index of a parameter.
- [AAX_Result GetParameterIDFromIndex](#) ([int32_t](#) iControllIndex, [AAX_IString](#) *oParameterIDString) const [AAX_OVERRIDE](#)
CALL: Retrieves the ID of a parameter.
- [AAX_Result GetParameterValueInfo](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) iSelector, [int32_t](#) *oValue) const [AAX_OVERRIDE](#)
CALL: Retrieves a property of a parameter.

Parameter setters and getters

These methods are used by the AAX host and by the plug-in's UI to retrieve and modify the values of the plug-in's parameters.

Note

The parameter setters in this section may generate asynchronous requests.

- [AAX_Result GetParameterValueFromString](#) ([AAX_CParamID](#) iParameterID, double *oValue, const [AAX_IString](#) &iValueString) const [AAX_OVERRIDE](#)
CALL: Converts a value string to a value.
- [AAX_Result GetParameterStringFromValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const [AAX_OVERRIDE](#)
CALL: Converts a normalized parameter value into a string representing its corresponding real value.
- [AAX_Result GetParameterValueString](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const [AAX_OVERRIDE](#)
CALL: Retrieves the value string associated with a parameter's current value.
- [AAX_Result GetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double *oValuePtr) const [AAX_OVERRIDE](#)
CALL: Retrieves a parameter's current value.
- [AAX_Result SetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue) [AAX_OVERRIDE](#)
CALL: Sets the specified parameter to a new value.
- [AAX_Result SetParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue) [AAX_OVERRIDE](#)
CALL: Sets the specified parameter to a new value relative to its current value.

Automated parameter helpers

These methods are used to lock and unlock automation system 'resources' when updating automatable parameters.

Note

You should never need to override these methods to extend their behavior beyond what is provided in [AAX_CEffectParameters](#) and [AAX_IParameter](#)

- [AAX_Result TouchParameter](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)
"Touches" (locks) a parameter in the automation system to a particular control in preparation for updates
- [AAX_Result ReleaseParameter](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)
Releases a parameter from a "touched" state.
- [AAX_Result UpdateParameterTouch](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iTouchState) [AAX_OVERRIDE](#)
Sets a "touched" state on a parameter.

Asynchronous parameter update methods

These methods are called by the AAX host when parameter values have been updated. They are called by the host and can be triggered by other plug-in modules via calls to [AAX_IParameter](#)'s `SetValue` methods, e.g. [SetValueWithFloat\(\)](#)

These methods are responsible for updating parameter values.

Do not call these methods directly! To ensure proper synchronization and to avoid problematic dependency chains, other methods (e.g. [SetParameterNormalizedValue\(\)](#)) and components (e.g. [AAX_IEffectGUI](#)) should always call a `SetValue` method on [AAX_IParameter](#) to update parameter values. The `SetValue` method will properly manage automation locks and other system resources.

- [AAX_Result UpdateParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_EUpdateSource](#) iSource) [AAX_OVERRIDE](#)
Updates a single parameter's state to its current value.
- [AAX_Result UpdateParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue) [AAX_OVERRIDE](#)
Updates a single parameter's state to its current value, as a difference with the parameter's previous value.
- [AAX_Result GenerateCoefficients](#) (void) [AAX_OVERRIDE](#)
Generates and dispatches new coefficient packets.

State reset handlers

- [AAX_Result ResetFieldData](#) ([AAX_CFieldIndex](#) inFieldIndex, void *oData, uint32_t inDataSize) const [AAX_OVERRIDE](#)
Called by the host to reset a private data field in the plug-in's algorithm.

Chunk methods

These methods are used to save and restore collections of plug-in state information, known as chunks. Chunks are used by the host when saving or restoring presets and session settings and when providing "compare" functionality for plug-ins.

The default implementation of these methods in [AAX_CEffectParameters](#) supports a single chunk that includes state information for all of the plug-in's registered parameters. Override all of these methods to add support for additional chunks in your plug-in, for example if your plug-in contains any persistent state that is not encapsulated by its set of registered parameters.

For reference, see also:

- [AAX_CChunkDataParser](#)
- [AAX_SPlugInChunk](#)
- [AAX_Result GetNumberOfChunks](#) (int32_t *oNumChunks) const [AAX_OVERRIDE](#)
Retrieves the number of chunks used by this plug-in.
- [AAX_Result GetChunkIDFromIndex](#) (int32_t iIndex, [AAX_CTypeID](#) *oChunkID) const [AAX_OVERRIDE](#)
Retrieves the ID associated with a chunk index.
- [AAX_Result GetChunkSize](#) ([AAX_CTypeID](#) iChunkID, uint32_t *oSize) const [AAX_OVERRIDE](#)

- Get the size of the data structure that can hold all of a chunk's information.
- [AAX_Result GetChunk](#) (AAX_CTypeID iChunkID, AAX_SPlugInChunk *oChunk) const [AAX_OVERRIDE](#)
Fills a block of data with chunk information representing the plug-in's current state.
- [AAX_Result SetChunk](#) (AAX_CTypeID iChunkID, const AAX_SPlugInChunk *iChunk) [AAX_OVERRIDE](#)
Restores a set of plug-in parameters based on chunk information.
- [AAX_Result CompareActiveChunk](#) (const AAX_SPlugInChunk *iChunkP, AAX_CBoolean *oIsEqual) const [AAX_OVERRIDE](#)
Determine if a chunk represents settings that are equivalent to the plug-in's current state.
- [AAX_Result GetNumberOfChanges](#) (int32_t *oNumChanges) const [AAX_OVERRIDE](#)
Retrieves the number of parameter changes made since the plug-in's creation.

Threads

Threading functions

- [AAX_Result TimerWakeup](#) () [AAX_OVERRIDE](#)
Periodic wakeup callback for idle-time operations.

Auxiliary UI methods

Methods defining the presentation of the plug-in on auxiliary UIs such as control surfaces

- [AAX_Result GetCurveData](#) (AAX_CTypeID iCurveType, const float *iValues, uint32_t iNumValues, float *oValues) const [AAX_OVERRIDE](#)
Generate a set of output values based on a set of given input values.
- [AAX_Result GetCurveDataMeterIds](#) (AAX_CTypeID iCurveType, uint32_t *oXMeterId, uint32_t *oYMeterId) const [AAX_OVERRIDE](#)
Indicates which meters correspond to the X and Y axes of the EQ or Dynamics graph.
- [AAX_Result GetCurveDataDisplayRange](#) (AAX_CTypeID iCurveType, float *oXMin, float *oXMax, float *oYMin, float *oYMax) const [AAX_OVERRIDE](#)
Determines the range of the graph shown by the plug-in.
- [AAX_Result UpdatePageTable](#) (uint32_t inTableType, int32_t inTablePageSize, IACFUnknown *iHostUnknown, IACFUnknown *ioPageTableUnknown) const [AAX_OVERRIDE](#) [AAX_FINAL](#)
Allow the plug-in to update its page tables.

Custom Data Methods

These functions exist as a proxiable way to move data between different modules (e.g. [AAX_IEffectParameters](#) and [AAX_IEffectGUI](#).) Using these, the GUI can query any data through [GetCustomData\(\)](#) with a plug-in defined *typeID*, *void** and *size*. This has an advantage over just sharing memory in that this function can work as a remote proxy as we enable those sorts of features later in the platform. Likewise, the GUI can also set arbitrary data on the data model by using the [SetCustomData\(\)](#) function with the same idea.

Note

These are plug-in internal only. They are not called from the host right now, or likely ever.

- [AAX_Result GetCustomData](#) (AAX_CTypeID iDataBlockID, uint32_t inDataSize, void *oData, uint32_t *oDataWritten) const [AAX_OVERRIDE](#)
An optional interface hook for getting custom data from another module.
- [AAX_Result SetCustomData](#) (AAX_CTypeID iDataBlockID, uint32_t inDataSize, const void *iData) [AAX_OVERRIDE](#)
An optional interface hook for setting custom data for use by another module.

MIDI methods

- [AAX_Result DoMIDITransfers](#) () [AAX_OVERRIDE](#)
MIDI update callback.
- [AAX_Result UpdateMIDINodes](#) (AAX_CFieldIndex inFieldIndex, AAX_CMidiPacket &iPacket) [AAX_OVERRIDE](#)

- *MIDI update callback.*
 • [AAX_Result UpdateControlMIDINodes](#) ([AAX_CTypeID](#) nodeID, [AAX_CMidiPacket](#) &iPacket) [AAX_OVERRIDE](#)
MIDI update callback for control MIDI nodes.

Hybrid audio methods

- [AAX_Result RenderAudio_Hybrid](#) ([AAX_SHybridRenderInfo](#) *ioRenderInfo) [AAX_OVERRIDE](#)
Hybrid audio render function.

Private data accessors

- [AAX_IController](#) * [Controller](#) ()
Access to the Effect controller.
- const [AAX_IController](#) * [Controller](#) () const
const access to the Effect controller
- [AAX_ITransport](#) * [Transport](#) ()
Access to the Transport object.
- const [AAX_ITransport](#) * [Transport](#) () const
const access to the Transport object
- [AAX_IAutomationDelegate](#) * [AutomationDelegate](#) ()
Access to the Effect's automation delegate.
- const [AAX_IAutomationDelegate](#) * [AutomationDelegate](#) () const
const access to the Effect's automation delegate

Public Member Functions inherited from [AAX_IEffectParameters](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid
- [AAX_DELETE](#) ([AAX_IEffectParameters](#) &operator=(const [AAX_IEffectParameters](#) &))

Auxiliary UI methods

Auxiliary UI methods

Hybrid audio methods

MIDI methods

Initialization and uninitialization

AAX host and plug-in event notification

Parameter information

These methods are used by the AAX host to retrieve information about the plug-in's data model.

For information about adding parameters to the plug-in and otherwise modifying the plug-in's data model, see [AAX_CParameterManager](#). For information about parameters, see [AAX_IParameter](#).

Parameter setters and getters

These methods are used by the AAX host and by the plug-in's UI to retrieve and modify the values of the plug-in's parameters.

Note

The parameter setters in this section may generate asynchronous requests.

Automated parameter helpers

These methods are used to lock and unlock automation system 'resources' when updating automatable parameters.

Note

You should never need to override these methods to extend their behavior beyond what is provided in [AAX_CEffectParameters](#) and [AAX_IParameter](#)

Asynchronous parameter update methods

These methods are called by the AAX host when parameter values have been updated. They are called by the host and can be triggered by other plug-in modules via calls to [AAX_IParameter](#)'s `SetValue` methods, e.g. [SetValueWithFloat\(\)](#)

These methods are responsible for updating parameter values.

Do not call these methods directly! To ensure proper synchronization and to avoid problematic dependency chains, other methods (e.g. [SetParameterNormalizedValue\(\)](#)) and components (e.g. [AAX_IEffectGUI](#)) should always call a `SetValue` method on [AAX_IParameter](#) to update parameter values. The `SetValue` method will properly manage automation locks and other system resources.

State reset handlers**Chunk methods**

These methods are used to save and restore collections of plug-in state information, known as chunks. Chunks are used by the host when saving or restoring presets and session settings and when providing "compare" functionality for plug-ins.

The default implementation of these methods in [AAX_CEffectParameters](#) supports a single chunk that includes state information for all of the plug-in's registered parameters. Override all of these methods to add support for additional chunks in your plug-in, for example if your plug-in contains any persistent state that is not encapsulated by its set of registered parameters.

Warning

Remember that plug-in chunk data may be loaded on a different platform from the one where it is saved. All data structures in the chunk must be properly data-aligned for compatibility across all platforms that the plug-in supports. See [AAX_ALIGN_FILE_ALG](#) for notes about common cross-platform pitfalls for data structure alignment.

For reference, see also:

- [AAX_CChunkDataParser](#)
- [AAX_SPlugInChunk](#)

Thread methods**Auxiliary UI methods****Custom data methods**

These functions exist as a proxiable way to move data between different modules (e.g. [AAX_IEffectParameters](#) and [AAX_IEffectGUI](#).) Using these, the GUI can query any data through [GetCustomData\(\)](#) with a plug-in defined `typeID`, `void` and `size`. This has an advantage over just sharing memory in that this function can work as a remote proxy as we enable those sorts of features later in the platform. Likewise, the GUI can also set arbitrary data on the data model by using the [SetCustomData\(\)](#) function with the same idea.*

Note

These are plug-in internal only. They are not called from the host right now, or likely ever.

MIDI methods

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Protected Member Functions**Parameter management methods**

- [AAX_Result SetTaperDelegate](#) ([AAX_CParamID](#) iParameterID, [AAX_ITaperDelegateBase](#) &iTaperDelegate, bool iPreserveValue)
- [AAX_Result SetDisplayDelegate](#) ([AAX_CParamID](#) iParameterID, [AAX_IDisplayDelegateBase](#) &iDisplayDelegate)
- bool [IsParameterTouched](#) ([AAX_CParamID](#) iParameterID) const
- bool [IsParameterLinkReady](#) ([AAX_CParamID](#) inParameterID, [AAX_EUpdateSource](#) inSource) const

Convenience functions

These convenience functions provide quick access to various aspects of the default [AAX_CEffectParameters](#) implementation.

- int32_t [mNumPlugInChanges](#)
- int32_t [mChunkSize](#)
- [AAX_CChunkDataParser](#) [mChunkParser](#)
- int32_t [mNumChunkedParameters](#)
- [AAX_CPacketDispatcher](#) [mPacketDispatcher](#)
- [AAX_CParameterManager](#) [mParameterManager](#)
- std::set< std::string > [mFilteredParameters](#)
- virtual [AAX_Result EffectInit](#) (void)
Initialization helper routine. Called from [AAX_CEffectParameters::Initialize](#).
- virtual [AAX_Result UpdatePageTable](#) (uint32_t, int32_t, [AAX_IPageTable](#) &) const
- void [FilterParameterIDOnSave](#) ([AAX_CParamID](#) controlId)
CALL: Indicates the indices of parameters that should not be saved in the default [AAX_CEffectParameters](#) chunk.
- void [BuildChunkData](#) (void) const
Clears out the current chunk in Chunk Parser and adds all of the new values. Used by default implementations of [GetChunk\(\)](#) and [GetChunkSize\(\)](#).

Additional Inherited Members**Public Attributes inherited from [AAX_IEffectParameters](#)**

- void **ppvObjOut [override](#)

14.13.3 Constructor & Destructor Documentation

14.13.3.1 AAX_CEffectParameters()

```
AAX_CEffectParameters::AAX_CEffectParameters (
    void )
```

14.13.3.2 ~AAX_CEffectParameters()

```
AAX_CEffectParameters::~~AAX_CEffectParameters (
    void )
```

14.13.4 Member Function Documentation

14.13.4.1 operator=()

```
AAX_CEffectParameters & AAX_CEffectParameters::operator= (
    const AAX_CEffectParameters & other )
```

14.13.4.2 Initialize()

```
AAX_Result AAX_CEffectParameters::Initialize (
    IACFUnknown * iController ) [virtual]
```

Main data model initialization. Called when plug-in instance is first instantiated.

Note

Most plug-ins should override [AAX_CEffectParameters::EffectInit\(\)](#) rather than directly overriding this method

Parameters

in	<i>iController</i>	A versioned reference that resolves to an AAX_IController interface
----	--------------------	---

This default implementation calls [EffectInit\(\)](#). Only override [Initialize\(\)](#) when additional initialization steps must be performed prior to [EffectInit\(\)](#).

Implements [AAX_IACFEffParameters](#).

14.13.4.3 Uninitialize()

```
AAX_Result AAX_CEffectParameters::Uninitialize (
    void ) [virtual]
```

Main data model uninitialization.

Todo Docs: When exactly is [AAX_IACFEffParameters::Uninitialize\(\)](#) called, and under what conditions?

Implements [AAX_IACFEffParameters](#).

14.13.4.4 NotificationReceived()

```
AAX_Result AAX_CEffectParameters::NotificationReceived (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- some notifications are sent only to the plug-in GUI while other notifications are sent only to the plug-in data model. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

A plug-in may also dispatch custom notifications using [AAX_IController::SendNotification\(\)](#). Custom notifications will be posted back to the plug-in's other objects which support a [NotificationReceived\(\)](#) method (e.g. the GUI).

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implements [AAX_IACFEffParameters](#).

14.13.4.5 GetNumberOfParameters()

```
AAX_Result AAX_CEffectParameters::GetNumberOfParameters (
    int32_t * oNumControls ) const [virtual]
```

CALL: Retrieves the total number of plug-in parameters.

Parameters

out	<i>oNumControls</i>	The number of parameters in the plug-in's Parameter Manager
-----	---------------------	---

Implements [AAX_IACFEffectParameters](#).

14.13.4.6 GetMasterBypassParameter()

```
AAX_Result AAX_CEffectParameters::GetMasterBypassParameter (
    AAX_IString * oIDString ) const [virtual]
```

CALL: Retrieves the ID of the plug-in's Master Bypass parameter.

This is required if you want our master bypass functionality in the host to hook up to your bypass parameters.

Parameters

out	<i>oIDString</i>	The ID of the plug-in's Master Bypass control
-----	------------------	---

Implements [AAX_IACFEffectParameters](#).

14.13.4.7 GetParameterIsAutomatable()

```
AAX_Result AAX_CEffectParameters::GetParameterIsAutomatable (
    AAX_CParamID iParameterID,
    AAX_CBoolean * oAutomatable ) const [virtual]
```

CALL: Retrieves information about a parameter's automatable status.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oAutomatable</i>	True if the queried parameter is automatable, false if it is not

Implements [AAX_IACFEffectParameters](#).

14.13.4.8 GetParameterNumberOfSteps()

```
AAX_Result AAX_CEffectParameters::GetParameterNumberOfSteps (
    AAX_CParamID iParameterID,
    int32_t * oNumSteps ) const [virtual]
```

CALL: Retrieves the number of discrete steps for a parameter.

Note

The value returned for `oNumSteps` MUST be greater than zero. All other values will be considered an error by the host.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oNumSteps</i>	The number of steps for this parameter

Implements [AAX_IACFEffParameters](#).

14.13.4.9 GetParameterName()

```
AAX_Result AAX_CEffectParameters::GetParameterName (
    AAX_CParamID iParameterID,
    AAX_IString * oName ) const [virtual]
```

CALL: Retrieves the full name for a parameter.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oName</i>	Reference to an AAX_IString owned by the host. The plug-in must set this string equal to the parameter's full name.

Implements [AAX_IACFEffParameters](#).

14.13.4.10 GetParameterNameOfLength()

```
AAX_Result AAX_CEffectParameters::GetParameterNameOfLength (
    AAX_CParamID iParameterID,
    AAX_IString * oName,
    int32_t iNameLength ) const [virtual]
```

CALL: Retrieves an abbreviated name for a parameter.

In general, lengths of 3 through 8 and 31 should be specifically addressed.

Host Compatibility Notes In most cases, the AAX host will call [GetParameterName\(\)](#) or [GetParameterNameOfLength\(\)](#) to retrieve parameter names for display. However, when Pro Tools is retrieving a plug-in name for display on a control surface the XML data stored in the plug-in's page tables will be used in preference to values retrieved from these methods.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oName</i>	Reference to an AAX_IString owned by the host. The plug-in must set this string equal to an abbreviated name for the parameter, using <i>iNameLength</i> characters or fewer.
in	<i>iNameLength</i>	The maximum number of characters in <i>oName</i>

Implements [AAX_IACFEffParameters](#).

14.13.4.11 GetParameterDefaultNormalizedValue()

```
AAX_Result AAX_CEffectParameters::GetParameterDefaultNormalizedValue (
    AAX_CParamID iParameterID,
    double * oValue ) const [virtual]
```

CALL: Retrieves default value of a parameter.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValue</i>	The parameter's default value

Implements [AAX_IACFEffParameters](#).

14.13.4.12 SetParameterDefaultNormalizedValue()

```
AAX_Result AAX_CEffectParameters::SetParameterDefaultNormalizedValue (
    AAX_CParamID iParameterID,
    double iValue ) [virtual]
```

CALL: Sets the default value of a parameter.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being updated
out	<i>iValue</i>	The parameter's new default value

Todo THIS IS NOT CALLED FROM HOST. USEFUL FOR INTERNAL USE ONLY?

Implements [AAX_IACFEffectParameters](#).

14.13.4.13 GetParameterType()

```
AAX_Result AAX_CEffectParameters::GetParameterType (
    AAX_CParamID iParameterID,
    AAX_EParameterType * oParameterType ) const [virtual]
```

CALL: Retrieves the type of a parameter.

Todo The concept of parameter type needs more documentation

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oParameterType</i>	The parameter's type

Implements [AAX_IACFEffectParameters](#).

14.13.4.14 GetParameterOrientation()

```
AAX_Result AAX_CEffectParameters::GetParameterOrientation (
    AAX_CParamID iParameterID,
    AAX_EParameterOrientation * oParameterOrientation ) const [virtual]
```

CALL: Retrieves the orientation that should be applied to a parameter's controls.

Todo update this documentation

This method allows you to specify the orientation of knob controls that are managed by the host (e.g. knobs on an attached control surface.)

Here is an example override of this method that reverses the orientation of a control for a parameter:

```
// AAX_IParameter* myBackwardsParameter
if (iParameterID == myBackwardsParameter->Identifier())
{
    *oParameterType =
        AAX_eParameterOrientation_BottomMinTopMax |
        AAX_eParameterOrientation_LeftMinRightMax |
        AAX_eParameterOrientation_RotaryWrapMode |
        AAX_eParameterOrientation_RotaryLeftMinRightMax;
}
```

The orientation options are set according to [AAX_EParameterOrientationBits](#)

Legacy Porting Notes [AAX_IEffectParameters::GetParameterOrientation\(\)](#) corresponds to the `GetControlOrientation()` method in the legacy RTAS/TDM SDK.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oParameterOrientation</i>	The orientation of the parameter

Implements [AAX_IACFEffParameters](#).

14.13.4.15 GetParameter()

```
AAX_Result AAX_CEffectParameters::GetParameter (
    AAX_CParamID iParameterID,
    AAX_IParameter ** oParameter ) [virtual]
```

CALL: Retrieves an arbitrary setting within a parameter.

This is a convenience function for accessing the richer parameter interface from the plug-in's other modules.

Note

This function must not be called by the host; [AAX_IParameter](#) is not safe for passing across the binary boundary with the host!

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oParameter</i>	A pointer to the returned parameter

Implements [AAX_IACFEffParameters](#).

14.13.4.16 GetParameterIndex()

```
AAX_Result AAX_CEffectParameters::GetParameterIndex (
    AAX_CParamID iParameterID,
    int32_t * oControlIndex ) const [virtual]
```

CALL: Retrieves the index of a parameter.

Although parameters are normally referenced by their `AAX_CParamID`, each parameter is also associated with a unique numeric index.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oControlIndex</i>	The parameter's numeric index

Implements [AAX_IACFEffectParameters](#).

14.13.4.17 GetParameterIDFromIndex()

```
AAX_Result AAX_CEffectParameters::GetParameterIDFromIndex (
    int32_t iControlIndex,
    AAX_IString * oParameterIDString ) const [virtual]
```

CALL: Retrieves the ID of a parameter.

This method can be used to convert a parameter's unique numeric index to its AAX_CParamID

Parameters

in	<i>iControlIndex</i>	The numeric index of the parameter that is being queried
out	<i>oParameterIDString</i>	Reference to an AAX_IString owned by the host. The plug-in must set this string equal to the parameter's ID.

Implements [AAX_IACFEffectParameters](#).

14.13.4.18 GetParameterValueInfo()

```
AAX_Result AAX_CEffectParameters::GetParameterValueInfo (
    AAX_CParamID iParameterID,
    int32_t iSelector,
    int32_t * oValue ) const [virtual]
```

CALL: Retrieves a property of a parameter.

This is a general purpose query that is specialized based on the value of *iSelector*. The currently supported selector values are described by [AAX_EParameterValueInfoSelector](#). The meaning of *oValue* is dependent upon *iSelector*.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
in	<i>iSelector</i>	The selector of the parameter value to retrieve. See AAX_EParameterValueInfoSelector
out	<i>oValue</i>	The value of the specified parameter

Implements [AAX_IACFEffectParameters](#).

14.13.4.19 GetParameterValueFromString()

```
AAX_Result AAX_CEffectParameters::GetParameterValueFromString (
    AAX_CParamID iParameterID,
```

```
double * oValue,
const AAX_IString & iValueString ) const [virtual]
```

CALL: Converts a value string to a value.

This method uses the queried parameter's display delegate and taper to convert a `char*` string into its corresponding value. The formatting of valueString must be supported by the parameter's display delegate in order for this call to succeed.

Legacy Porting Notes This method corresponds to `CProcess::MapControlStringToVal()` in the RTAS/TDM SDK

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValue</i>	The value associated with valueString
in	<i>iValueString</i>	The formatted value string that will be converted into a value

Implements [AAX_IACFEffParameters](#).

14.13.4.20 GetParameterStringFromValue()

```
AAX_Result AAX_CEffectParameters::GetParameterStringFromValue (
    AAX_CParamID iParameterID,
    double iValue,
    AAX_IString * oValueString,
    int32_t iMaxLength ) const [virtual]
```

CALL: Converts a normalized parameter value into a string representing its corresponding real value.

This method uses the queried parameter's display delegate and taper to convert a normalized value into the corresponding `char*` value string for its real value.

Legacy Porting Notes This method corresponds to `CProcess::MapControlValToString()` in the RTAS/TDM SDK

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
in	<i>iValue</i>	The normalized value that will be converted to a formatted valueString
out	<i>oValueString</i>	The formatted value string associated with value
in	<i>iMaxLength</i>	The maximum length of valueString

Implements [AAX_IACFEffParameters](#).

14.13.4.21 GetParameterValueString()

```
AAX_Result AAX_CEffectParameters::GetParameterValueString (
    AAX_CParamID iParameterID,
    AAX_IString * oValueString,
    int32_t iMaxLength ) const [virtual]
```

CALL: Retrieves the value string associated with a parameter's current value.

This method uses the queried parameter's display delegate and taper to convert its current value into a corresponding `char*` value string.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValueString</i>	The formatted value string associated with the parameter's current value
in	<i>iMaxLength</i>	The maximum length of valueString

Implements [AAX_IACFEffectParameters](#).

14.13.4.22 GetParameterNormalizedValue()

```
AAX_Result AAX_CEffectParameters::GetParameterNormalizedValue (
    AAX_CParamID iParameterID,
    double * oValuePtr ) const [virtual]
```

CALL: Retrieves a parameter's current value.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValuePtr</i>	The parameter's current value

Implements [AAX_IACFEffectParameters](#).

14.13.4.23 SetParameterNormalizedValue()

```
AAX_Result AAX_CEffectParameters::SetParameterNormalizedValue (
    AAX_CParamID iParameterID,
    double iValue ) [virtual]
```

CALL: Sets the specified parameter to a new value.

[SetParameterNormalizedValue\(\)](#) is responsible for initiating any process that is required in order to update all of the parameter's controls (e.g. in the plug-in's GUI, on control surfaces, in automation lanes, etc.) In most cases, the parameter manager will handle this initiation step.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being set
in	<i>iValue</i>	The value to which the parameter should be set

Implements [AAX_IACFEffParameters](#).

14.13.4.24 SetParameterNormalizedRelative()

```
AAX_Result AAX_CEffectParameters::SetParameterNormalizedRelative (
    AAX_CParamID iParameterID,
    double iValue ) [virtual]
```

CALL: Sets the specified parameter to a new value relative to its current value.

This method is used in cases when a relative control value is more convenient, for example when updating a GUI control using a mouse wheel or the arrow keys. Note that the host may apply the parameter's step size prior to calling [SetParameterNormalizedRelative\(\)](#) in order to determine the correct value for aValue.

[SetParameterNormalizedRelative\(\)](#) can be used to incorporate "wrapping" behavior in a parameter's controls, if desired. If this behavior is not desired, then this method must properly account for overflow of the parameter's normalized value.

[SetParameterNormalizedRelative\(\)](#) is responsible for initiating any process that is required in order to update all of the parameter's controls (e.g. in the plug-in's GUI, on control surfaces, in automation lanes, etc.) In most cases, the parameter manager will handle this initiation step.

See also [UpdateParameterNormalizedRelative\(\)](#).

Todo REMOVE THIS METHOD (?)

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
in	<i>iValue</i>	The change in value that should be applied to the parameter

Todo NOT CURRENTLY CALLED FROM THE HOST. USEFUL FOR INTERNAL USE ONLY?

Implements [AAX_IACFEffParameters](#).

14.13.4.25 TouchParameter()

```
AAX_Result AAX_CEffectParameters::TouchParameter (
    AAX_CParamID iParameterID ) [virtual]
```

"Touches" (locks) a parameter in the automation system to a particular control in preparation for updates

This method is called by the Parameter Manager to prime a parameter for receiving new automation data. When an automatable parameter is touched by a control, it will reject input from other controls until it is released.

Note

You should never need to override this method when using [AAX_CEffectParameters](#).

Parameters

in	<i>iParameterID</i>	The parameter that is being touched
----	---------------------	-------------------------------------

Implements [AAX_IACFEffectParameters](#).

14.13.4.26 ReleaseParameter()

```
AAX_Result AAX_CEffectParameters::ReleaseParameter (
    AAX_CParamID iParameterID ) [virtual]
```

Releases a parameter from a "touched" state.

This method is called by the Parameter Manager to release a parameter so that any control may send updates to the parameter.

Note

You should never need to override this method when using [AAX_CEffectParameters](#).

Parameters

in	<i>iParameterID</i>	The parameter that is being released
----	---------------------	--------------------------------------

Implements [AAX_IACFEffectParameters](#).

14.13.4.27 UpdateParameterTouch()

```
AAX_Result AAX_CEffectParameters::UpdateParameterTouch (
    AAX_CParamID iParameterID,
    AAX_CBoolean iTouchState ) [virtual]
```

Sets a "touched" state on a parameter.

Note

This method should be overridden when dealing with linked parameters. Do NOT use this method to keep track of touch states. Use the [automation delegate](#) for that.

Parameters

in	<i>iParameterID</i>	The parameter that is changing touch states.
in	<i>iTouchState</i>	The touch state of the parameter.

Implements [AAX_IACFEffEffectParameters](#).

14.13.4.28 UpdateParameterNormalizedValue()

```
AAX_Result AAX_CEffectParameters::UpdateParameterNormalizedValue (
    AAX_CParamID iParameterID,
    double iValue,
    AAX_EUpdateSource iSource ) [virtual]
```

Updates a single parameter's state to its current value.

Note

Do *not* call this method from the plug-in. This method should be called by the host only. To set parameter values from within the plug-in, use the [AAX_IParameter](#) interface.

Todo FLAGGED FOR CONSIDERATION OF REVISION

Parameters

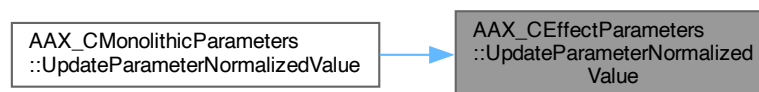
in	<i>iParameterID</i>	The ID of the parameter that is being updated
in	<i>iValue</i>	The parameter's current value, to which its internal state must be updated
in	<i>iSource</i>	The source of the update

Implements [AAX_IACFEffEffectParameters](#).

Reimplemented in [AAX_CMonolithicParameters](#).

Referenced by [AAX_CMonolithicParameters::UpdateParameterNormalizedValue\(\)](#).

Here is the caller graph for this function:



14.13.4.29 UpdateParameterNormalizedRelative()

```
AAX_Result AAX_CEffectParameters::UpdateParameterNormalizedRelative (
    AAX_CParamID iParameterID,
    double iValue ) [virtual]
```

Updates a single parameter's state to its current value, as a difference with the parameter's previous value.

Deprecated This is not called from the host. It *may* still be useful for internal calls within the plug-in, though it should only ever be used to update non-automatable parameters. Automatable parameters should always be updated through the [AAX_IParameter](#) interface, which will ensure proper coordination with other automation clients.

[UpdateParameterNormalizedRelative\(\)](#) can be used to incorporate "wraparound" behavior in a parameter's controls, if desired. If this behavior is not desired, then this method must properly account for overflow of the parameter's normalized value.

See also

[SetParameterNormalizedRelative\(\)](#)

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being updated
in	<i>iValue</i>	The difference between the parameter's current value and its previous value (normalized). The parameter's state must be updated to reflect this difference.

Implements [AAX_IACFEffectParameters](#).

14.13.4.30 GenerateCoefficients()

```
AAX_Result AAX_CEffectParameters::GenerateCoefficients (
    void ) [virtual]
```

Generates and dispatches new coefficient packets.

This method is responsible for updating the coefficient packets associated with all parameters whose states have changed since the last call to [GenerateCoefficients\(\)](#). The host may call this method once for every parameter update, or it may "batch" parameter updates such that changes for several parameters are all handled by a single call to [GenerateCoefficients\(\)](#).

For more information on tracking parameters' statuses using the [AAX_CPacketDispatcher](#), helper class, see [AAX_CPacketDispatcher::SetDirty\(\)](#).

Note

Do *not* call this method from the plug-in. This method should be called by the host only. To set parameter values from within the plug-in, use the [AAX_IParameter](#) interface.

Implements [AAX_IACFEffectParameters](#).

Reimplemented in [AAX_CMonolithicParameters](#).

Referenced by [AAX_CMonolithicParameters::GenerateCoefficients\(\)](#).

Here is the caller graph for this function:



14.13.4.31 ResetFieldData()

```

AAX_Result AAX_CEffectParameters::ResetFieldData (
    AAX_CFieldIndex inFieldIndex,
    void * oData,
    uint32_t inDataSize ) const [virtual]
  
```

Called by the host to reset a private data field in the plug-in's algorithm.

This method is called sequentially for all private data fields on Effect initialization and during any "reset" event, such as priming for a non-real-time render. This method is called before the algorithm's optional initialization callback, and the initialized private data will be available within that callback via its context block.

See also

[Algorithm initialization.](#)

Warning

Any data structures that will be passed between platforms (for example, sent to a TI DSP in an AAX DSP plug-in) must be properly data-aligned for compatibility across both platforms. See [AAX_ALIGN_FILE_ALG](#) for more information about guaranteeing cross-platform compatibility of data structures used for algorithm processing.

Parameters

in	<i>inFieldIndex</i>	The index of the field that is being initialized
out	<i>oData</i>	The pre-allocated block of data that should be initialized
in	<i>inDataSize</i>	The size of the data block, in bytes

Implements [AAX_IACFEffParameters](#).

Reimplemented in [AAX_CMonolithicParameters](#).

Referenced by [AAX_CMonolithicParameters::ResetFieldData\(\)](#).

Here is the caller graph for this function:



14.13.4.32 GetNumberOfChunks()

```
AAX_Result AAX_CEffectParameters::GetNumberOfChunks (
    int32_t * oNumChunks ) const [virtual]
```

Retrieves the number of chunks used by this plug-in.

Parameters

out	<i>oNumChunks</i>	The number of distinct chunks used by this plug-in
-----	-------------------	--

Implements [AAX_IACFEffParameters](#).

14.13.4.33 GetChunkIDFromIndex()

```
AAX_Result AAX_CEffectParameters::GetChunkIDFromIndex (
    int32_t iIndex,
    AAX_CTypeID * oChunkID ) const [virtual]
```

Retrieves the ID associated with a chunk index.

Parameters

in	<i>iIndex</i>	Index of the queried chunk
out	<i>oChunkID</i>	ID of the queried chunk

Implements [AAX_IACFEffParameters](#).

14.13.4.34 GetChunkSize()

```
AAX_Result AAX_CEffectParameters::GetChunkSize (
    AAX_CTypeID iChunkID,
    uint32_t * oSize ) const [virtual]
```

Get the size of the data structure that can hold all of a chunk's information.

If *chunkID* is one of the plug-in's custom chunks, initialize **size* to the size of the chunk's data in bytes.

This method is invoked every time a chunk is saved, therefore it is possible to have dynamically sized chunks. However, note that each call to [GetChunkSize\(\)](#) will correspond to a following call to [GetChunk\(\)](#). The chunk provided in [GetChunk\(\)](#) *must* have the same size as the *size* provided by [GetChunkSize\(\)](#).

Legacy Porting Notes In [AAX](#), the value provided by [GetChunkSize\(\)](#) should *NOT* include the size of the chunk header. The value should *ONLY* reflect the size of the chunk's data.

Parameters

in	<i>iChunkID</i>	ID of the queried chunk
out	<i>oSize</i>	The chunk's size in bytes

Implements [AAX_IACFEffEffectParameters](#).

14.13.4.35 GetChunk()

```
AAX_Result AAX_CEffectParameters::GetChunk (
    AAX_CTypeID iChunkID,
    AAX_SPlugInChunk * oChunk ) const [virtual]
```

Fills a block of data with chunk information representing the plug-in's current state.

By calling this method, the host is requesting information about the current state of the plug-in. The following chunk fields should be explicitly populated in this method. Other fields will be populated by the host.

- [AAX_SPlugInChunk::fData](#)
- [AAX_SPlugInChunk::fVersion](#)
- [AAX_SPlugInChunk::fName](#) (Optional)
- [AAX_SPlugInChunk::fSize](#) (Data size only)

Warning

Remember that this chunk data may be loaded on a different platform from the one where it is saved. All data structures in the chunk must be properly data-aligned for compatibility across all platforms that the plug-in supports. See [AAX_ALIGN_FILE_ALG](#) for notes about common cross-platform pitfalls for data structure alignment.

Parameters

in	<i>iChunkID</i>	ID of the chunk that should be provided
out	<i>oChunk</i>	A preallocated block of memory that should be populated with the chunk's data.

Implements [AAX_IACFEffectParameters](#).

14.13.4.36 SetChunk()

```
AAX_Result AAX_CEffectParameters::SetChunk (
    AAX_CTypeID iChunkID,
    const AAX_SPlugInChunk * iChunk ) [virtual]
```

Restores a set of plug-in parameters based on chunk information.

By calling this method, the host is attempting to update the plug-in's current state to match the data stored in a chunk. The plug-in should initialize itself to this new state by calling [SetParameterNormalizedValue\(\)](#) for each of the relevant parameters.

Parameters

in	<i>iChunkID</i>	ID of the chunk that is being set
in	<i>iChunk</i>	The chunk

Implements [AAX_IACFEffectParameters](#).

14.13.4.37 CompareActiveChunk()

```
AAX_Result AAX_CEffectParameters::CompareActiveChunk (
    const AAX_SPlugInChunk * iChunkP,
    AAX_CBoolean * oIsEqual ) const [virtual]
```

Determine if a chunk represents settings that are equivalent to the plug-in's current state.

Host Compatibility Notes In Pro Tools, this method will only be called if a prior call to [GetNumberOfChanges\(\)](#) has indicated that the plug-in's state has changed. If the plug-in's current settings are different from the settings in *aChunkP* then the plug-in's Compare Light will be illuminated in the plug-in header, allowing users to toggle between the plug-in's custom state and its saved state.

Parameters

in	<i>iChunkP</i>	The chunk that is to be tested
out	<i>oIsEqual</i>	True if the chunk represents equivalent settings when compared with the plug-in's current state. False if the chunk represents non-equivalent settings

Implements [AAX_IACFEffectParameters](#).

14.13.4.38 GetNumberOfChanges()

```
AAX_Result AAX_CEffectParameters::GetNumberOfChanges (
    int32_t * oNumChanges ) const [virtual]
```

Retrieves the number of parameter changes made since the plug-in's creation.

This method is polled regularly by the host, and can additionally be triggered by some events such as mouse clicks. When the number provided by this method changes, the host subsequently calls [CompareActiveChunk\(\)](#) to determine if the plug-in's Compare light should be activated.

The value provided by this method should increment with each call to [UpdateParameterNormalizedValue\(\)](#)

Parameters

out	<i>oNumChanges</i>	Must be set to indicate the number of parameter changes that have occurred since plug-in initialization.
-----	--------------------	--

Implements [AAX_IACFEffectParameters](#).

14.13.4.39 TimerWakeup()

```
AAX_Result AAX_CEffectParameters::TimerWakeup ( ) [virtual]
```

Periodic wakeup callback for idle-time operations.

This method is called from the host using a non-main thread. In general, it should be driven at approximately one call per 30 ms. However, the wakeup is not guaranteed to be called at any regular interval - for example, it could be held off by a high real-time processing load - and there is no host contract regarding maximum latency between wakeup calls.

This wakeup thread runs continuously and cannot be armed/disarmed or by the plug-in.

Implements [AAX_IACFEffectParameters](#).

Reimplemented in [AAX_CMonolithicParameters](#).

Referenced by [AAX_CMonolithicParameters::TimerWakeup\(\)](#).

Here is the caller graph for this function:



14.13.4.40 GetCurveData()

```
AAX_Result AAX_CEffectParameters::GetCurveData (
    AAX_CTypeID iCurveType,
    const float * iValues,
    uint32_t iNumValues,
    float * oValues ) const [virtual]
```

Generate a set of output values based on a set of given input values.

This method is used by the host to generate graphical curves. Given a set of input values, e.g. frequencies in Hz, this method should generate a corresponding set of output values, e.g. dB gain at each frequency. The semantics of these input and output values are dictated by [iCurveType](#). See [AAX_ECurveType](#).

Plug-ins may also define custom curve type IDs to use this method internally. For example, the plug-in's GUI could use this method to request curve data in an arbitrary format.

Note

- This method may be called by the host simultaneously from multiple threads with different *iValues*.

Note

- *oValues* must be allocated by caller with the same size as *iValues* (*iNumValues*).

Host Compatibility Notes Versions of S6 software which support the [GetCurveDataDisplayRange\(\)](#) method will not display a plug-in's curve data unless both [GetCurveData\(\)](#) and [GetCurveDataDisplayRange\(\)](#) are supported by the plug-in.

Warning

S6 currently polls this method to update a plug-in's EQ or dynamics curves based on changes to the parameters mapped to the plug-in's EQ or dynamics center section page tables. Parameters that are not included in these page tables will not trigger updates to the curves displayed on S6. (GWSW-7314, [PTSW-195316 / PT-218485](#))

Parameters

in	<i>iCurveType</i>	One of AAX_ECurveType
in	<i>iValues</i>	An array of input values
in	<i>iNumValues</i>	The size of <i>iValues</i>
out	<i>oValues</i>	An array of output values

Returns

This method must return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not support curve data for the requested *iCurveType*

Implements [AAX_IACFEfectParameters](#).

14.13.4.41 GetCurveDataMeterIds()

```
AAX_Result AAX_CEffectParameters::GetCurveDataMeterIds (
    AAX_CTypeID iCurveType,
    uint32_t * oXMeterId,
    uint32_t * oYMeterId ) const [virtual]
```

Indicates which meters correspond to the X and Y axes of the EQ or Dynamics graph.

These meters can be used by attached control surfaces to present an indicator in the same X/Y coordinate plane as the plug-in's curve data.

Parameters

in	<i>iCurveType</i>	One of AAX_ECurveType
out	<i>oXMeterId</i>	Id of the X-axis meter
out	<i>oYMeterId</i>	Id of the Y-axis meter

Returns

This method should return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not implement it.

Implements [AAX_IACFEffectParameters_V3](#).

14.13.4.42 GetCurveDataDisplayRange()

```
AAX_Result AAX_CEffectParameters::GetCurveDataDisplayRange (
    AAX_CTypeID iCurveType,
    float * oXMin,
    float * oXMax,
    float * oYMin,
    float * oYMax ) const [virtual]
```

Determines the range of the graph shown by the plug-in.

Min/max arguments define the range of the axes of the graph.

Parameters

in	<i>iCurveType</i>	One of AAX_ECurveType
out	<i>oXMin</i>	Min value of X-axis range
out	<i>oXMax</i>	Max value of X-axis range
out	<i>oYMin</i>	Min value of Y-axis range
out	<i>oYMax</i>	Max value of Y-axis range

Returns

This method should return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not implement it.

Implements [AAX_IACFEffectParameters_V3](#).

14.13.4.43 UpdatePageTable() [1/2]

```
AAX_Result AAX_CEffectParameters::UpdatePageTable (
    uint32_t inTableType,
    int32_t inTablePageSize,
    IACFUnknown * iHostUnknown,
    IACFUnknown * ioPageTableUnknown ) const [virtual]
```

Allow the plug-in to update its page tables.

Called by the plug-in host, usually in response to a [AAX_eNotificationEvent_ParameterMappingChanged](#) notification sent from the plug-in.

Use this method to change the page table mapping for the plug-in instance or to apply other changes to auxiliary UIs which use the plug-in page tables, such as setting focus to a new page.

See [Page Table Guide](#) for more information about page tables.

Parameters

in	<i>inTableType</i>	Four-char type identifier for the table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the table
in	<i>iHostUnknown</i>	Unknown interface from the host which may support interfaces providing additional features or information. All interfaces queried from this unknown will be valid only within the scope of this UpdatePageTable() execution and will be relevant for only the current plug-in instance.
in, out	<i>ioPageTableUnknown</i>	Unknown interface which supports AAX_IPageTable . This object represents the page table data which is currently stored by the host for this plug-in instance for the given table type and page size. This data and may be edited within the scope of UpdatePageTable() to change the page table mapping for this plug-in instance.

Returns

This method should return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not implement it or when no change is requested by the plug-in. This allows optimizations to be used in the host when no UI update is required following this call.

See also

[AAX_eNotificationEvent_ParameterMappingChanged](#)

Note

For convenience, do not override this method. Instead, override the [protected overload](#) which provides a prepared copy of the relevant [AAX_IPageTable](#) host interface.

Implements [AAX_IACFEffectParameters_V4](#).

14.13.4.44 GetCustomData()

```
AAX_Result AAX_CEffectParameters::GetCustomData (
    AAX_CTypeID iDataBlockID,
    uint32_t inDataSize,
    void * oData,
    uint32_t * oDataWritten ) const [virtual]
```

An optional interface hook for getting custom data from another module.

Parameters

in	<i>iDataBlockID</i>	Identifier for the requested block of custom data
in	<i>inDataSize</i>	Size of provided buffer, in bytes
out	<i>oData</i>	Pointer to an allocated buffer. Data will be written here.
out	<i>oDataWritten</i>	The number of bytes actually written

Implements [AAX_IACFEffParameters](#).

14.13.4.45 SetCustomData()

```
AAX_Result AAX_CEffectParameters::SetCustomData (
    AAX_CTypeID iDataBlockID,
    uint32_t inDataSize,
    const void * iData ) [virtual]
```

An optional interface hook for setting custom data for use by another module.

Parameters

in	<i>iDataBlockID</i>	Identifier for the provided block of custom data
in	<i>inDataSize</i>	Size of provided buffer, in bytes
in	<i>iData</i>	Pointer to the data buffer

Implements [AAX_IACFEffParameters](#).

14.13.4.46 DoMIDITransfers()

```
AAX_Result AAX_CEffectParameters::DoMIDITransfers ( ) [inline], [virtual]
```

MIDI update callback.

Call [AAX_IController::GetNextMIDIPacket\(\)](#) from within this method to retrieve and process MIDI packets directly within the Effect's data model. MIDI data will also be delivered to the Effect algorithm.

This method is called regularly by the host, similarly to [AAX_IEffectParameters::TimerWakeup\(\)](#)

Implements [AAX_IACFEffParameters](#).

References [AAX_SUCCESS](#).

14.13.4.47 UpdateMIDINodes()

```
AAX_Result AAX_CEffectParameters::UpdateMIDINodes (
    AAX_CFieldIndex inFieldIndex,
    AAX_CMidiPacket & iPacket ) [virtual]
```

MIDI update callback.

This method is called by the host for each pending MIDI packet for MIDI nodes in algorithm context structure. Overwrite this method in Plug-In's EffectParameter class if you want to receive MIDI data packets directly in the data model. MIDI data will also be delivered to the Effect algorithm.

The host calls this method in Effects that register one or more MIDI nodes using [AAX_IComponentDescriptor::AddMIDINode\(\)](#). Effects that do not require MIDI data to be sent to the plug-in algorithm should override [UpdateControlMIDINodes\(\)](#).

Parameters

in	<i>inFieldIndex</i>	MIDI node field index in algorithm context structure
in	<i>iPacket</i>	The incoming MIDI packet for the node

Implements [AAX_IACFEffectParameters_V2](#).

14.13.4.48 UpdateControlMIDINodes()

```
AAX_Result AAX_CEffectParameters::UpdateControlMIDINodes (
    AAX_CTypeID nodeID,
    AAX_CMidiPacket & iPacket ) [virtual]
```

MIDI update callback for control MIDI nodes.

This method is called by the host for each pending MIDI packet for Control MIDI nodes. Overwrite this method in Plug-In's EffectParameter class if you want to receive MIDI data packets directly in the data model.

The host calls this method in Effects that register one or more Control MIDI nodes using [AAX_IEffectDescriptor::AddControlMIDINode\(\)](#). Effects with algorithms that use MIDI data nodes should override [UpdateMIDINodes\(\)](#).

Note

This method will not be called if an Effect includes any MIDI nodes in its algorithm context structure.

Parameters

in	<i>nodeID</i>	Identifier for the MIDI node
in	<i>iPacket</i>	The incoming MIDI packet for the node

Implements [AAX_IACFEffectParameters_V2](#).

14.13.4.49 RenderAudio_Hybrid()

```
AAX_Result AAX_CEffectParameters::RenderAudio_Hybrid (
    AAX_SHybridRenderInfo * ioRenderInfo ) [virtual]
```

Hybrid audio render function.

This method is called from the host to render audio for the hybrid piece of the algorithm.

Note

To use this method plug-in should register some hybrid inputs and outputs in "Describe"

Implements [AAX_IACFEffectParameters_V2](#).

14.13.4.50 Controller() [1/2]

```
AAX_IController * AAX_CEffectParameters::Controller ( )
```

Access to the Effect controller.

Referenced by [AAX_CMonolithicParameters::GenerateCoefficients\(\)](#).

Here is the caller graph for this function:

**14.13.4.51 Controller() [2/2]**

```
const AAX_IController * AAX_CEffectParameters::Controller ( ) const
```

const access to the Effect controller

14.13.4.52 Transport() [1/2]

```
AAX_ITransport * AAX_CEffectParameters::Transport ( )
```

Access to the Transport object.

14.13.4.53 Transport() [2/2]

```
const AAX_ITransport * AAX_CEffectParameters::Transport ( ) const
```

const access to the Transport object

14.13.4.54 AutomationDelegate() [1/2]

```
AAX_IAutomationDelegate * AAX_CEffectParameters::AutomationDelegate ( )
```

Access to the Effect's automation delegate.

14.13.4.55 AutomationDelegate() [2/2]

```
const AAX_IAutomationDelegate * AAX_CEffectParameters::AutomationDelegate ( ) const
```

const access to the Effect's automation delegate

14.13.4.56 SetTaperDelegate()

```
AAX_Result AAX_CEffectParameters::SetTaperDelegate (
    AAX_CParamID iParameterID,
    AAX_ITaperDelegateBase & iTaperDelegate,
    bool iPreserveValue ) [protected]
```

14.13.4.57 SetDisplayDelegate()

```
AAX_Result AAX_CEffectParameters::SetDisplayDelegate (
    AAX_CParamID iParameterID,
    AAX_IDisplayDelegateBase & iDisplayDelegate ) [protected]
```

14.13.4.58 IsParameterTouched()

```
bool AAX_CEffectParameters::IsParameterTouched (
    AAX_CParamID iParameterID ) const [protected]
```

14.13.4.59 IsParameterLinkReady()

```
bool AAX_CEffectParameters::IsParameterLinkReady (
    AAX_CParamID inParameterID,
    AAX_EUpdateSource inSource ) const [protected]
```

14.13.4.60 EffectInit()

```
virtual AAX_Result AAX_CEffectParameters::EffectInit (
    void ) [inline], [protected], [virtual]
```

Initialization helper routine. Called from [AAX_CEffectParameters::Initialize](#).

Override to add parameters, packets, meters, and to do any other custom initialization.

Add custom parameters:

- Create an [AAX_CParameter](#) for each parameter in the plug-in
- Call [AAX_CParameterManager::AddParameter\(\)](#) using mParameterManager to add parameters to the Parameter Manager

Note

See bug [AAXSDK-897](#)

Register packets:

- Call [AAX_CPacketDispatcher::RegisterPacket\(\)](#) using mPacketDispatcher to register a packet and handling callback.

References [AAX_SUCCESS](#).

14.13.4.61 UpdatePageTable() [2/2]

```
virtual AAX_Result AAX_CEffectParameters::UpdatePageTable (
    uint32_t ,
    int32_t ,
    AAX_IPageTable & ) const [inline], [protected], [virtual]
```

Protected overload of [UpdatePageTable\(\)](#)

Override this version of the method for convenience. This allows the default [UpdatePageTable\(\)](#) implementation to handle the interface conversion from [IACFUnknown](#) to [AAX_IPageTable](#).

Returns

This method should return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not implement it or when no change is made by the plug-in. This allows optimizations to be used in the host when no UI update is required following this call.

References [AAX_ERROR_UNIMPLEMENTED](#).

14.13.4.62 FilterParameterIDOnSave()

```
void AAX_CEffectParameters::FilterParameterIDOnSave (
    AAX_CParamID controlId ) [protected]
```

CALL: Indicates the indices of parameters that should not be saved in the default [AAX_CEffectParameters](#) chunk.

Allows specific parameters to filtered out of the default [AAX_CEffectParameters](#) "Save Settings" functionality. This call is automatically invoked on the Master Bypass control when specified by the DefineMasterBypassControlIndex() call.

Parameters

in	<i>controlID</i>	The ID of the parameter that should be removed from the default chunk
----	------------------	---

14.13.4.63 BuildChunkData()

```
void AAX_CEffectParameters::BuildChunkData (
    void ) const [protected]
```

Clears out the current chunk in Chunk Parser and adds all of the new values. Used by default implementations of [GetChunk\(\)](#) and [GetChunkSize\(\)](#).

14.13.5 Member Data Documentation**14.13.5.1 mNumPluginChanges**

```
int32_t AAX_CEffectParameters::mNumPluginChanges [protected]
```

14.13.5.2 mChunkSize

```
int32_t AAX_CEffectParameters::mChunkSize [mutable], [protected]
```

14.13.5.3 mChunkParser

```
AAX_CChunkDataParser AAX_CEffectParameters::mChunkParser [mutable], [protected]
```

14.13.5.4 mNumChunkedParameters

```
int32_t AAX_CEffectParameters::mNumChunkedParameters [protected]
```

14.13.5.5 mPacketDispatcher

```
AAX_CPacketDispatcher AAX_CEffectParameters::mPacketDispatcher [protected]
```

14.13.5.6 mParameterManager

```
AAX_CParameterManager AAX_CEffectParameters::mParameterManager [protected]
```

Referenced by [AAX_CMonolithicParameters::UpdateParameterNormalizedValue\(\)](#).

14.13.5.7 mFilteredParameters

```
std::set<std::string> AAX_CEffectParameters::mFilteredParameters [protected]
```

The documentation for this class was generated from the following file:

- [AAX_CEffectParameters.h](#)

14.14 AAX_CheckedResult Class Reference

```
#include <AAX_Exception.h>
```

14.14.1 Description

Error checker convenience class for [AAX_Result](#)

Implicitly convertible to an [AAX_Result](#).

Provides an overloaded `operator=()` which will throw an [AAX::Exception::ResultError](#) if assigned a non-success result.

Warning

Never use this class outside of an exception catch scope

If the host supports [AAX_TRACE](#) tracing, a log is emitted when the exception is thrown. A stacktrace is added if the host's trace priority filter level is set to [kAAX_Trace_Priority_Lowest](#)

When an error is encountered, [AAX_CheckedResult](#) throws an [AAX_CheckedResult::Exception](#) exception and clears its internal result value.

```
#include "AAX_Exception.h"
AAX_Result SomeCheckedMethod()
{
    AAX_Result result = AAX_SUCCESS;
    try {
        AAX_CheckedResult cr;
        cr = ResultFunc1();
        cr = ResultFunc2();
    }
    catch (const AAX_CheckedResult::Exception& ex)
    {
        // handle exception; do not rethrow
        result = ex.Result();
    }
    catch (...)
    {
        result = AAX_ERROR_UNKNOWN_EXCEPTION;
    }

    return result;
}
```

Note

The [AAX](#) Library method which calls [GetEffectDescriptions\(\)](#) on the plug-in includes an appropriate exception handler, so [AAX_CheckedResult](#) objects may be used within a plug-in's describe code without additional catch scopes.

```
#include "AAX_Exception.h"
AAX_Result GetEffectDescriptions( AAX_ICollection * outCollection )
{
    AAX_CheckedResult cr;
    cr = MyDescriptionSubroutine1();
    cr = outCollection->AddEffect(...);
    // etc.
    return cr;
}
```

It is assumed that the exception handler will resolve any error state and that the [AAX_CheckedResult](#) may therefore continue to be used from a clean state following the exception catch block.

If the previous error value is required then it can be retrieved using [AAX_CheckedResult::LastError\(\)](#).

```
// in this example, the exception is handled and
// success is returned from MyFunc1()
AAX_Result MyFunc1()
{
    AAX_CheckedResult cr;

    try {
        cr = MethodThatReturnsError();
    } catch (const AAX::Exception::ResultError& ex) {
        // exception is fully handled here
    }

    // cr now holds a success value
    return cr;
}

// in this example, MyFunc2() returns the first
// non-successful value which was encountered
AAX_Result MyFunc2()
{
    AAX_CheckedResult cr;

    try {
        AAX_SWALLOW(cr = MethodThatMayReturnError1());
        AAX_SWALLOW(cr = MethodThatMayReturnError2());
        cr = MethodThatMayReturnError3();
    } catch (const AAX::Exception::ResultError& ex) {
```

```

    // exception might not be fully handled
}

// pass the last error on to the caller
return cr.LastError();
}

```

It is possible to add one or more accepted non-success values to an [AAX_CheckedResult](#) so that these values will not trigger exceptions:

```

AAX_CheckedResult cr;
try {
    cr.AddAcceptedResult(AcceptableErrCode);
    cr = MethodThatReturnsAcceptedError();
    cr = MethodThatReturnsAnotherError();
} catch (const AAX::Exception::ResultError& ex) {
    // handle the exception
}

```

Public Types

- typedef [AAX::Exception::ResultError](#) Exception

Public Member Functions

- [~AAX_CheckedResult](#) ()
- [AAX_CheckedResult](#) ()
Construct an [AAX_CheckedResult](#) in a success state.
- [AAX_CheckedResult](#) ([AAX_Result](#) inResult)
Implicit conversion constructor from [AAX_Result](#).
- void [AddAcceptedResult](#) ([AAX_Result](#) inResult)
Add an expected result which will not result in a throw.
- void [ResetAcceptedResults](#) ()
- [AAX_CheckedResult](#) & [operator=](#) ([AAX_Result](#) inResult)
Assignment to [AAX_Result](#).
- [AAX_CheckedResult](#) & [operator|=](#) ([AAX_Result](#) inResult)
bitwise-or assignment to [AAX_Result](#)
- [operator AAX_Result](#) () const
Conversion to [AAX_Result](#).
- void [Clear](#) ()
Clears the current result state.
- [AAX_Result](#) [LastError](#) () const
Get the last non-success result which was stored in this object, or AAX_SUCCESS if no non-success result was ever stored in this object.

14.14.2 Member Typedef Documentation

14.14.2.1 Exception

```
typedef AAX::Exception::ResultError AAX_CheckedResult::Exception
```

14.14.3 Constructor & Destructor Documentation

14.14.3.1 ~AAX_CheckedResult()

```
AAX_CheckedResult::~~AAX_CheckedResult ( ) [inline]
```

14.14.3.2 AAX_CheckedResult() [1/2]

```
AAX_CheckedResult::AAX_CheckedResult ( ) [inline]
```

Construct an [AAX_CheckedResult](#) in a success state.

14.14.3.3 AAX_CheckedResult() [2/2]

```
AAX_CheckedResult::AAX_CheckedResult (
    AAX\_Result inResult ) [inline]
```

Implicit conversion constructor from [AAX_Result](#).

Implicit conversion is OK in order to support [AAX_CheckedResult](#) cr = SomeFunc()

14.14.4 Member Function Documentation

14.14.4.1 AddAcceptedResult()

```
void AAX_CheckedResult::AddAcceptedResult (
    AAX\_Result inResult ) [inline]
```

Add an expected result which will not result in a throw.

It is acceptable for some methods to return certain non-success values such as [AAX_RESULT_PACKET_STREAM_NOT_EMPTY](#) or [AAX_RESULT_NEW_PACKET_POSTED](#)

14.14.4.2 ResetAcceptedResults()

```
void AAX_CheckedResult::ResetAcceptedResults ( ) [inline]
```

References [AAX_SUCCESS](#).

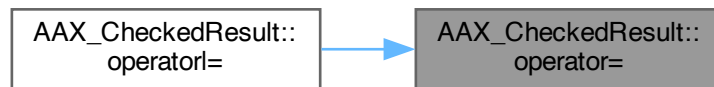
14.14.4.3 operator=()

```
AAX_CheckedResult & AAX_CheckedResult::operator= (
    AAX_Result inResult ) [inline]
```

Assignment to [AAX_Result](#).

Referenced by [operator|=\(\(\)\)](#).

Here is the caller graph for this function:



14.14.4.4 operator" |=(())

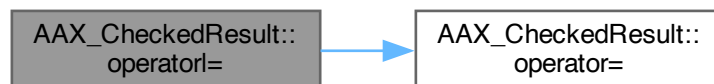
```
AAX_CheckedResult & AAX_CheckedResult::operator|=(
    AAX_Result inResult ) [inline]
```

bitwise-or assignment to [AAX_Result](#)

Sometimes used in legacy code to aggregate results into a single `AAX_Result` value

References [operator=\(\)](#).

Here is the call graph for this function:



14.14.4.5 operator AAX_Result()

```
AAX_CheckedResult::operator AAX_Result ( ) const [inline]
```

Conversion to [AAX_Result](#).

14.14.4.6 Clear()

```
void AAX_CheckedResult::Clear ( ) [inline]
```

Clears the current result state.

Does not affect the set of accepted results

References [AAX_SUCCESS](#).

14.14.4.7 LastError()

```
AAX_Result AAX_CheckedResult::LastError ( ) const [inline]
```

Get the last non-success result which was stored in this object, or [AAX_SUCCESS](#) if no non-success result was ever stored in this object.

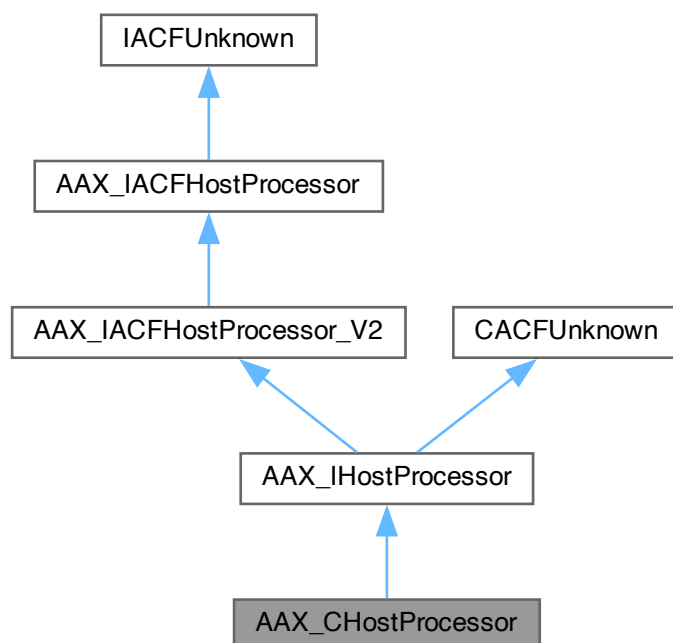
The documentation for this class was generated from the following file:

- [AAX_Exception.h](#)

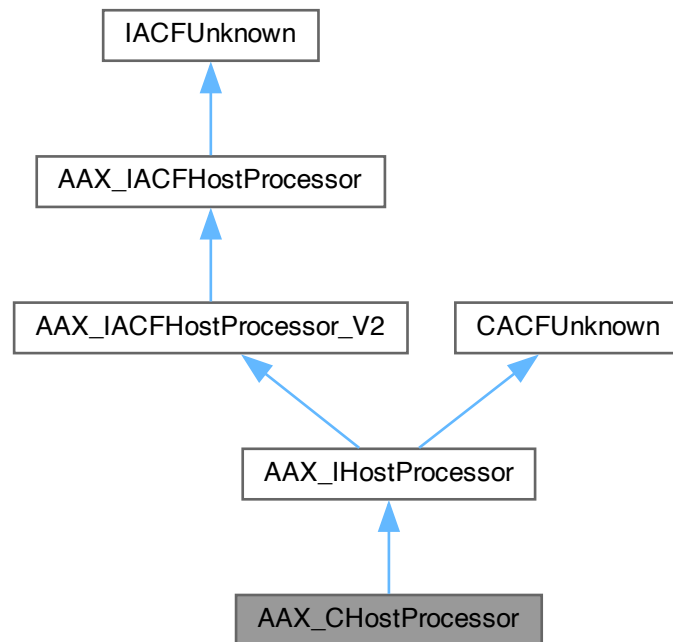
14.15 AAX_CHostProcessor Class Reference

```
#include <AAX_CHostProcessor.h>
```

Inheritance diagram for AAX_CHostProcessor:



Collaboration diagram for AAX_CHostProcessor:



14.15.1 Description

Concrete implementation of the [AAX_IHostProcessor](#) interface for non-real-time processing.

Host processor objects are used to process regions of audio data in a non-real-time context.

- Host processors must generate output samples linearly and incrementally, but may randomly access samples from the processing region on the timeline for input. See [GetAudio\(\)](#) for more information.
- Host processors may re-define the processing region using [AAX_CHostProcessor::TranslateOutputBounds\(\)](#).

See also

[AAX_IHostProcessorDelegate](#)

Public Member Functions

- [AAX_CHostProcessor](#) (void)
- virtual [~AAX_CHostProcessor](#) ()

Initialization and uninitialization

- [AAX_Result Initialize](#) (IACFUnknown *iController) [AAX_OVERRIDE](#)

- Host Processor initialization.*
- [AAX_Result Uninitialize \(\) AAX_OVERRIDE](#)
- Host Processor teardown.*

Host processor interface

- [AAX_Result InitOutputBounds](#) (int64_t iSrcStart, int64_t iSrcEnd, int64_t *oDstStart, int64_t *oDstEnd) [AAX_OVERRIDE](#)
Sets the processing region.
- [AAX_Result SetLocation](#) (int64_t iSample) [AAX_OVERRIDE](#)
Updates the relative sample location of the current processing frame.
- [AAX_Result RenderAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, float *const iAudioOuts[], int32_t iAudioOutCount, int32_t *ioWindowSize) [AAX_OVERRIDE](#)
Perform the signal processing.
- [AAX_Result PreRender](#) (int32_t inAudioInCount, int32_t iAudioOutCount, int32_t iWindowSize) [AAX_OVERRIDE](#)
Invoked right before the start of a Preview or Render pass.
- [AAX_Result PostRender](#) () [AAX_OVERRIDE](#)
Invoked at the end of a Render pass.
- [AAX_Result AnalyzeAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int32_t *ioWindowSize) [AAX_OVERRIDE](#)
Override this method if the plug-in needs to analyze the audio prior to a Render pass.
- [AAX_Result PreAnalyze](#) (int32_t inAudioInCount, int32_t iWindowSize) [AAX_OVERRIDE](#)
Invoked right before the start of an Analysis pass.
- [AAX_Result PostAnalyze](#) () [AAX_OVERRIDE](#)
Invoked at the end of an Analysis pass.
- [AAX_Result GetClipNameSuffix](#) (int32_t inMaxLength, [AAX_IString](#) *outString) const [AAX_OVERRIDE](#)
Called by host application to retrieve a custom string to be appended to the clip name.

Public Member Functions inherited from [AAX_IHostProcessor](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfIID](#) &riid
- [AAX_DELETE](#) ([AAX_IHostProcessor](#) &operator=(const [AAX_IHostProcessor](#) &))
- virtual [AAX_Result GetClipNameSuffix](#) (int32_t inMaxLength, [AAX_IString](#) *outString) const =0
Called by host application to retrieve a custom string to be appended to the clip name.
- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Host Processor initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Host Processor teardown.
- virtual [AAX_Result InitOutputBounds](#) (int64_t iSrcStart, int64_t iSrcEnd, int64_t *oDstStart, int64_t *oDstEnd)=0
Sets the processing region.
- virtual [AAX_Result SetLocation](#) (int64_t iSample)=0
Updates the relative sample location of the current processing frame.
- virtual [AAX_Result RenderAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, float *const iAudioOuts[], int32_t iAudioOutCount, int32_t *ioWindowSize)=0
Perform the signal processing.
- virtual [AAX_Result PreRender](#) (int32_t inAudioInCount, int32_t iAudioOutCount, int32_t iWindowSize)=0
Invoked right before the start of a Preview or Render pass.
- virtual [AAX_Result PostRender](#) ()=0
Invoked at the end of a Render pass.

- virtual [AAX_Result AnalyzeAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int32_t *ioWindow↵Size)=0
Override this method if the plug-in needs to analyze the audio prior to a Render pass.
- virtual [AAX_Result PreAnalyze](#) (int32_t inAudioInCount, int32_t iWindowSize)=0
Invoked right before the start of an Analysis pass.
- virtual [AAX_Result PostAnalyze](#) ()=0
Invoked at the end of an Analysis pass.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const acfIID &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Convenience methods

- [AAX_IEffectParameters](#) * [GetEffectParameters](#) ()
- const [AAX_IEffectParameters](#) * [GetEffectParameters](#) () const
- [AAX_IHostProcessorDelegate](#) * [GetHostProcessorDelegate](#) ()
- const [AAX_IHostProcessorDelegate](#) * [GetHostProcessorDelegate](#) () const
- int64_t [GetLocation](#) () const
The relative sample location of the current processing frame.
- int64_t [GetInputRange](#) () const
The length (in samples) of the current timeline selection.
- int64_t [GetOutputRange](#) () const
The length (in samples) of the clip that will be rendered to the timeline.
- int64_t [GetSrcStart](#) () const
The sample position of the beginning of the current timeline selection relative to the beginning of the current input selection, i.e. 0.
- int64_t [GetSrcEnd](#) () const
The sample position of the end of the current timeline selection relative to the beginning of the current input selection.
- int64_t [GetDstStart](#) () const
The sample position of the beginning of the of the clip that will be rendered to the timeline relative to the beginning of the current input selection.
- int64_t [GetDstEnd](#) () const
The sample position of the end of the of the clip that will be rendered to the timeline relative to the beginning of the current input selection.
- virtual [AAX_Result TranslateOutputBounds](#) (int64_t iSrcStart, int64_t iSrcEnd, int64_t &oDstStart, int64_t &oDstEnd)
Define the boundaries of the clip that will be rendered to the timeline.
- virtual [AAX_Result GetAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int64_t inLocation, int32_t *ioNumSamples)
Randomly access audio from the timeline.
- virtual int32_t [GetSideChainInputNum](#) ()
CALL: Returns the index of the side chain input buffer.
- [AAX_IController](#) * [Controller](#) ()
- const [AAX_IController](#) * [Controller](#) () const
- [AAX_IHostProcessorDelegate](#) * [HostProcessorDelegate](#) ()
- const [AAX_IHostProcessorDelegate](#) * [HostProcessorDelegate](#) () const
- [AAX_IEffectParameters](#) * [EffectParameters](#) ()
- const [AAX_IEffectParameters](#) * [EffectParameters](#) () const

Additional Inherited Members

Public Attributes inherited from [AAX_IHostProcessor](#)

- void **ppvObjOut [override](#)

14.15.2 Constructor & Destructor Documentation

14.15.2.1 AAX_CHostProcessor()

```
AAX_CHostProcessor::AAX_CHostProcessor (
    void )
```

14.15.2.2 ~AAX_CHostProcessor()

```
virtual AAX_CHostProcessor::~~AAX_CHostProcessor ( ) [virtual]
```

14.15.3 Member Function Documentation

14.15.3.1 Initialize()

```
AAX_Result AAX_CHostProcessor::Initialize (
    IACFUnknown * iController ) [virtual]
```

Host Processor initialization.

Parameters

in	<i>iController</i>	A versioned reference that can be resolved to both an AAX_IController interface and an AAX_IHostProcessorDelegate
----	--------------------	---

Implements [AAX_IACFHostProcessor](#).

14.15.3.2 Uninitialize()

```
AAX_Result AAX_CHostProcessor::Uninitialize ( ) [virtual]
```

Host Processor teardown.

Implements [AAX_IACFHostProcessor](#).

14.15.3.3 InitOutputBounds()

```
AAX_Result AAX_CHostProcessor::InitOutputBounds (
    int64_t iSrcStart,
    int64_t iSrcEnd,
    int64_t * oDstStart,
    int64_t * oDstEnd ) [virtual]
```

Sets the processing region.

This method allows offline processing plug-ins to vary the length and/or start/end points of the audio processing region.

This method is called in a few different scenarios:

- Before an analyze, process or preview of data begins.
- At the end of every preview loop.
- After the user makes a new data selection on the timeline.

Plug-ins that inherit from [AAX_CHostProcessor](#) should not override this method. Instead, use the following convenience functions:

- To retrieve the length or boundaries of the processing region, use [GetInputRange\(\)](#), [GetSrcStart\(\)](#), etc.
- To change the boundaries of the processing region before processing begins, use [AAX_CHostProcessor::TranslateOutputBounds\(\)](#).

Note

Currently, a host processor may not randomly access samples outside of the boundary defined by `oDstStart` and `oDstEnd`.

Legacy Porting Notes DAE no longer makes use of the `mStartBound` and `mEndBounds` member variables that existed in the legacy RTAS/TDM SDK. Use `oDstStart` and `oDstEnd` instead (preferably by overriding [TranslateOutputBounds\(\)](#).)

Parameters

in	<i>iSrcStart</i>	The selection start of the user selected region. This is will always return 0 for a given selection on the timeline.
in	<i>iSrcEnd</i>	The selection end of the user selected region. This will always return the value of the selection length on the timeline.
in	<i>oDstStart</i>	The starting sample location in the output audio region. By default, this is the same as <i>iSrcStart</i> .
in	<i>oDstEnd</i>	The ending sample location in the output audio region. By default, this is the same as <i>iSrcEnd</i> .

Implements [AAX_IACFHostProcessor](#).

14.15.3.4 SetLocation()

```
AAX_Result AAX_CHostProcessor::SetLocation (
    int64_t iSample ) [virtual]
```

Updates the relative sample location of the current processing frame.

This method is called by the host to update the relative sample location of the current processing frame.

Note

Plug-ins should not override this method; instead, use [AAX_CHostProcessor::GetLocation\(\)](#) to retrieve the current relative sample location.

Parameters

in	<i>iSample</i>	The sample location of the first sample in the current processing frame relative to the beginning of the full processing buffer
----	----------------	---

Implements [AAX_IACFHostProcessor](#).

14.15.3.5 RenderAudio()

```
AAX_Result AAX_CHostProcessor::RenderAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    float *const iAudioOuts[],
    int32_t iAudioOutCount,
    int32_t * ioWindowSize ) [virtual]
```

Perform the signal processing.

This method is called by the host to invoke the plug-in's signal processing.

Legacy Porting Notes This method is a replacement for the AudioSuite `ProcessAudio` method

Parameters

in	<i>inAudioIns</i>	Input audio buffer
in	<i>inAudioInCount</i>	The number of input channels
in	<i>iAudioOuts</i>	The number of output channels
in	<i>iAudioOutCount</i>	A user defined destination end of the ingested audio
in	<i>ioWindowSize</i>	Window buffer length of the received audio

Implements [AAX_IACFHostProcessor](#).

14.15.3.6 PreRender()

```
AAX_Result AAX_CHostProcessor::PreRender (
    int32_t  inAudioInCount,
    int32_t  iAudioOutCount,
    int32_t  iWindowSize ) [virtual]
```

Invoked right before the start of a Preview or Render pass.

This method is called by the host to allow a plug-in to make any initializations before processing actually begins. Upon a Preview pass, PreRender will also be called at the beginning of every "loop".

See also

[AAX_eProcessingState_StartPass](#), [AAX_eProcessingState_BeginPassGroup](#)

Parameters

in	<i>inAudioInCount</i>	The number if input channels
in	<i>iAudioOutCount</i>	The number of output channels
in	<i>iWindowSize</i>	Window buffer length of the ingested audio

Implements [AAX_IACFHostProcessor](#).

14.15.3.7 PostRender()

```
AAX_Result AAX_CHostProcessor::PostRender ( ) [virtual]
```

Invoked at the end of a Render pass.

Note

Upon a Preview pass, PostRender will not be called until Preview has stopped.

See also

[AAX_eProcessingState_StopPass](#), [AAX_eProcessingState_EndPassGroup](#)

Implements [AAX_IACFHostProcessor](#).

14.15.3.8 AnalyzeAudio()

```
AAX_Result AAX_CHostProcessor::AnalyzeAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    int32_t * ioWindowSize ) [virtual]
```

Override this method if the plug-in needs to analyze the audio prior to a Render pass.

Use this after declaring the appropriate properties in Describe. See [AAX_eProperty_RequiresAnalysis](#) and [AAX_eProperty_OptionalAnalysis](#)

To request an analysis pass from within a plug-in, use [AAX_IHostProcessorDelegate::ForceAnalyze\(\)](#)

Legacy Porting Notes Ported from AudioSuite's `AnalyzeAudio(bool isMasterBypassed)` method

Parameters

in	<i>inAudioIns</i>	Input audio buffer
in	<i>inAudioInCount</i>	The number of input channels
in	<i>ioWindowSize</i>	Window buffer length of the ingested audio

Implements [AAX_IACFHostProcessor](#).

14.15.3.9 PreAnalyze()

```
AAX_Result AAX_CHostProcessor::PreAnalyze (
    int32_t inAudioInCount,
    int32_t iWindowSize ) [virtual]
```

Invoked right before the start of an Analysis pass.

This method is called by the host to allow a plug-in to make any initializations before an Analysis pass actually begins.

See also

[AAX_eProcessingState_StartPass](#), [AAX_eProcessingState_BeginPassGroup](#)

Parameters

in	<i>inAudioInCount</i>	The number if input channels
in	<i>iWindowSize</i>	Window buffer length of the ingested audio

Implements [AAX_IACFHostProcessor](#).

14.15.3.10 PostAnalyze()

```
AAX_Result AAX_CHostProcessor::PostAnalyze ( ) [virtual]
```

Invoked at the end of an Analysis pass.

Note

In Pro Tools, a long execution time for this method will hold off the main application thread and cause a visible hang. If the plug-in must perform any long running tasks before initiating processing then it is best to perform these tasks in [AAX_IHostProcessor::PreRender\(\)](#)

See also

[AAX_eProcessingState_StopPass](#), [AAX_eProcessingState_EndPassGroup](#)

Implements [AAX_IACFHostProcessor](#).

14.15.3.11 GetClipNameSuffix()

```
AAX_Result AAX_CHostProcessor::GetClipNameSuffix (
    int32_t inMaxLength,
    AAX_IString * outString ) const [virtual]
```

Called by host application to retrieve a custom string to be appended to the clip name.

If no string is provided then the host's default will be used.

Parameters

in	<i>inMaxLength</i>	The maximum allowed string length, not including the NULL terminating char
out	<i>outString</i>	Add a value to this string to provide a custom clip suffix

Implements [AAX_IACFHostProcessor_V2](#).

14.15.3.12 GetEffectParameters() [1/2]

```
AAX_IEffectParameters * AAX_CHostProcessor::GetEffectParameters ( ) [inline]
```

14.15.3.13 GetEffectParameters() [2/2]

```
const AAX_IEffectParameters * AAX_CHostProcessor::GetEffectParameters ( ) const [inline]
```

14.15.3.14 GetHostProcessorDelegate() [1/2]

```
AAX_IHostProcessorDelegate * AAX_CHostProcessor::GetHostProcessorDelegate ( ) [inline]
```

14.15.3.15 GetHostProcessorDelegate() [2/2]

```
const AAX_IHostProcessorDelegate * AAX_CHostProcessor::GetHostProcessorDelegate ( ) const [inline]
```

14.15.3.16 GetLocation()

```
int64_t AAX_CHostProcessor::GetLocation ( ) const [inline]
```

The relative sample location of the current processing frame.

This method returns the relative sample location for the current [RenderAudio\(\)](#) processing frame. For example, if a value of 10 is provided for the [RenderAudio\(\)](#) `ioWindow` parameter, then calls to this method from within each execution of [RenderAudio\(\)](#) will return 0, 10, 20,...

14.15.3.17 GetInputRange()

```
int64_t AAX_CHostProcessor::GetInputRange ( ) const [inline]
```

The length (in samples) of the current timeline selection.

14.15.3.18 GetOutputRange()

```
int64_t AAX_CHostProcessor::GetOutputRange ( ) const [inline]
```

The length (in samples) of the clip that will be rendered to the timeline.

14.15.3.19 GetSrcStart()

```
int64_t AAX_CHostProcessor::GetSrcStart ( ) const [inline]
```

The sample position of the beginning of the current timeline selection relative to the beginning of the current input selection, i.e. 0.

14.15.3.20 GetSrcEnd()

```
int64_t AAX_CHostProcessor::GetSrcEnd ( ) const [inline]
```

The sample position of the end of the current timeline selection relative to the beginning of the current input selection.

14.15.3.21 GetDstStart()

```
int64_t AAX_CHostProcessor::GetDstStart ( ) const [inline]
```

The sample position of the beginning of the of the clip that will be rendered to the timeline relative to the beginning of the current input selection.

This value will be equal to the value returned by [GetSrcStart\(\)](#) unless the selection boundaries have been modified by overriding [TranslateOutputBounds\(\)](#)

14.15.3.22 GetDstEnd()

```
int64_t AAX_CHostProcessor::GetDstEnd ( ) const [inline]
```

The sample position of the end of the of the clip that will be rendered to the timeline relative to the beginning of the current input selection.

This value will be equal to the value returned by [GetSrcStart\(\)](#) unless the selection boundaries have been modified by overriding [TranslateOutputBounds\(\)](#)

14.15.3.23 TranslateOutputBounds()

```
virtual AAX_Result AAX_CHostProcessor::TranslateOutputBounds (
    int64_t iSrcStart,
    int64_t iSrcEnd,
    int64_t & oDstStart,
    int64_t & oDstEnd ) [protected], [virtual]
```

Define the boundaries of the clip that will be rendered to the timeline.

This method is called from [AAX_CHostProcessor::InitOutputBounds\(\)](#), providing a convenient hook for re-defining the processing region boundaries. See [InitOutputBounds\(\)](#) for more information.

Parameters

in	<i>iSrcStart</i>	The selection start of the user selected region. This is will always return 0 for a given selection on the timeline.
in	<i>iSrcEnd</i>	The selection end of the user selected region. This will always return the value of the selection length on the timeline.
in	<i>oDstStart</i>	The starting sample location in the output audio region. By default, this is the same as <i>iSrcStart</i> .
in	<i>oDstEnd</i>	The ending sample location in the output audio region. By default, this is the same as <i>iSrcEnd</i> .

14.15.3.24 GetAudio()

```
virtual AAX_Result AAX_CHostProcessor::GetAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    int64_t inLocation,
    int32_t * ioNumSamples ) [protected], [virtual]
```

Randomly access audio from the timeline.

This is a convenience wrapper around [AAX_IHostProcessorDelegate::GetAudio\(\)](#).

Parameters

in	<i>inAudioIns</i>	Timeline audio buffer(s). This must be set to <i>inAudioIns</i> from AAX_IHostProcessor::RenderAudio()
in	<i>inAudioInCount</i>	Number of buffers in <i>inAudioIns</i> . This must be set to <i>inAudioInCount</i> from AAX_IHostProcessor::RenderAudio()
in	<i>inLocation</i>	A sample location relative to the beginning of the currently processed region, e.g. a value of 0 corresponds to the timeline location returned by AAX_CHostProcessor::GetSrcStart()
in, out	<i>ioNumSamples</i>	<ul style="list-style-type: none"> • Input: The maximum number of samples to read. • Output: The actual number of samples that were read from the timeline

14.15.3.25 GetSideChainInputNum()

```
virtual int32_t AAX_CHostProcessor::GetSideChainInputNum ( ) [protected], [virtual]
```

CALL: Returns the index of the side chain input buffer.

This is a convenience wrapper around [AAX_IHostProcessorDelegate::GetSideChainInputNum\(\)](#)

14.15.3.26 Controller() [1/2]

```
AAX_IController * AAX_CHostProcessor::Controller ( ) [inline], [protected]
```

14.15.3.27 Controller() [2/2]

```
const AAX_IController * AAX_CHostProcessor::Controller ( ) const [inline], [protected]
```

14.15.3.28 HostProcessorDelegate() [1/2]

```
AAX_IHostProcessorDelegate * AAX_CHostProcessor::HostProcessorDelegate ( ) [inline], [protected]
```

14.15.3.29 HostProcessorDelegate() [2/2]

```
const AAX_IHostProcessorDelegate * AAX_CHostProcessor::HostProcessorDelegate ( ) const [inline],  
[protected]
```

14.15.3.30 EffectParameters() [1/2]

```
AAX_IEffectParameters * AAX_CHostProcessor::EffectParameters ( ) [inline], [protected]
```

14.15.3.31 EffectParameters() [2/2]

```
const AAX_IEffectParameters * AAX_CHostProcessor::EffectParameters ( ) const [inline], [protected]
```

The documentation for this class was generated from the following file:

- [AAX_CHostProcessor.h](#)

14.16 AAX_CHostServices Class Reference

```
#include <AAX_CHostServices.h>
```

14.16.1 Description

Method access to a singleton implementation of the [AAX_IHostServices](#) interface.

Static Public Member Functions

- static void [Set](#) ([IACFUnknown](#) *pUnkHost)
- static [AAX_Result](#) [HandleAssertFailure](#) (const char *iFile, int32_t iLine, const char *iNote, int32_t iFlags=[AAX_eAssertFlags_Default](#))
Handle an assertion failure.
- static [AAX_Result](#) [Trace](#) ([AAX_ETracePriorityHost](#) iPriority, const char *iMessage,...)
Log a trace message.
- static [AAX_Result](#) [StackTrace](#) ([AAX_ETracePriorityHost](#) iTracePriority, [AAX_ETracePriorityHost](#) iStack←
TracePriority, const char *iMessage,...)
Log a trace message or a stack trace.

14.16.2 Member Function Documentation

14.16.2.1 Set()

```
static void AAX_CHostServices::Set (
    IACFUnknown * pUnkHost ) [static]
```

14.16.2.2 HandleAssertFailure()

```
static AAX_Result AAX_CHostServices::HandleAssertFailure (
    const char * iFile,
    int32_t iLine,
    const char * iNote,
    int32_t iFlags = AAX_eAssertFlags_Default ) [static]
```

Handle an assertion failure.

Use this method to delegate assertion failure handling to the host

Use `iFlags` to request that specific behavior be included when handling the failure. This request may not be fulfilled by the host, and absence of a flag does not preclude the host from using that behavior when handling the failure.

Parameters

in	<i>iFile</i>	The name of the file containing the assert check. Usually <code>__FILE__</code>
in	<i>iLine</i>	The line number of the assert check. Usually <code>__LINE__</code>
in	<i>iNote</i>	Text to display related to the assert. Usually the condition which failed
in	<i>iFlags</i>	Bitfield of AAX_EAssertFlags to request specific handling behavior

14.16.2.3 Trace()

```
static AAX_Result AAX_CHostServices::Trace (
    AAX_ETracePriorityHost iPriority,
    const char * iMessage,
    ... ) [static]
```

Log a trace message.

Parameters

in	<i>iPriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iMessage</i>	Message string to log

14.16.2.4 StackTrace()

```
static AAX_Result AAX_CHostServices::StackTrace (
    AAX_ETracePriorityHost iTracePriority,
    AAX_ETracePriorityHost iStackTracePriority,
    const char * iMessage,
    ... ) [static]
```

Log a trace message or a stack trace.

If the logging output filtering is set to include logs with `iStackTracePriority` then both the logging message and a stack trace will be emitted, regardless of `iTracePriority`.

If the logging output filtering is set to include logs with `iTracePriority` but to exclude logs with `iStackTracePriority` then this will emit a normal log with no stack trace.

Parameters

in	<i>iTracePriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iStackTracePriority</i>	Priority of the stack trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iMessage</i>	Message string to log

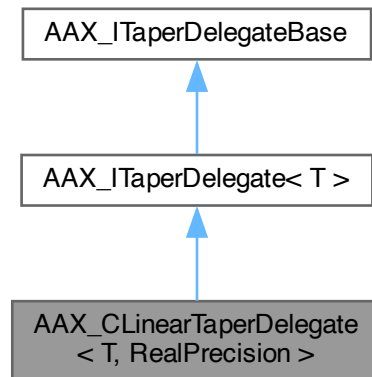
The documentation for this class was generated from the following file:

- [AAX_CHostServices.h](#)

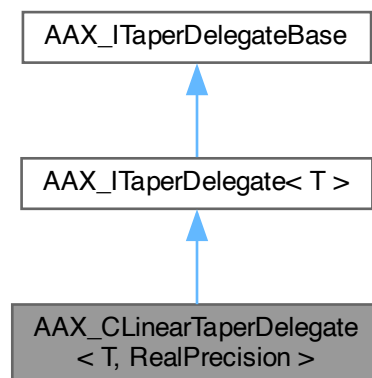
14.17 AAX_CLinearTaperDelegate< T, RealPrecision > Class Template Reference

```
#include <AAX_CLinearTaperDelegate.h>
```

Inheritance diagram for `AAX_CLinearTaperDelegate< T, RealPrecision >`:



Collaboration diagram for `AAX_CLinearTaperDelegate< T, RealPrecision >`:



14.17.1 Description

```
template<typename T, int32_t RealPrecision = 0>
class AAX_CLinearTaperDelegate< T, RealPrecision >
```

A linear taper conforming to [AAX_ITaperDelegate](#).

This taper spaces a parameter's real values evenly between its minimum and maximum, with a linear mapping between the parameter's real and normalized values.

RealPrecision

In addition to its type templization, this taper includes a precision template parameter. RealPrecision is a multiplier that works in conjunction with the round() function to limit the precision of the real values provided by this taper. For example, if RealPrecision is 1000, it will round to the closest 0.001 when doing any sort of value conversion. If RealPrecision is 1, it will round to the nearest integer. If RealPrecision is 1000000, it will round to the nearest 0.000001. This is particularly useful for preventing things like 1.9999999 truncating down to 1 instead of rounding up to 2.

To accomplish this behavior, the taper multiplies its unrounded parameter values by RealPrecision, rounds the result to the nearest valid value, then divides RealPrecision back out.

Rounding will be disabled if RealPrecision is set to a value less than 1. This is the default.

Public Member Functions

- [AAX_CLinearTaperDelegate](#) (T minValue=0, T maxValue=1)
Constructs a Linear Taper with specified minimum and maximum values.
- [AAX_CLinearTaperDelegate](#)< T, RealPrecision > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the taper delegate.
- T [GetMinimumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's minimum real value.
- T [GetMaximumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's maximum real value.
- T [ConstrainRealValue](#) (T value) const [AAX_OVERRIDE](#)
Applies a constraint to the value and returns the constrained value.
- T [NormalizedToReal](#) (double normalizedValue) const [AAX_OVERRIDE](#)
Converts a normalized value to a real value.
- double [RealToNormalized](#) (T realValue) const [AAX_OVERRIDE](#)
Normalizes a real parameter value.

- virtual [AAX_ITaperDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the taper delegate.
- virtual T [GetMaximumValue](#) () const =0
Returns the taper's maximum real value.
- virtual T [GetMinimumValue](#) () const =0
Returns the taper's minimum real value.
- virtual T [ConstrainRealValue](#) (T value) const =0
Applies a constraint to the value and returns the constrained value.
- virtual T [NormalizedToReal](#) (double normalizedValue) const =0
Converts a normalized value to a real value.
- virtual double [RealToNormalized](#) (T realValue) const =0
Normalizes a real parameter value.

Public Member Functions inherited from [AAX_ITaperDelegateBase](#)

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

Protected Member Functions

- `T Round (double iValue) const`

14.17.2 Constructor & Destructor Documentation

14.17.2.1 AAX_CLinearTaperDelegate()

```
template<typename T , int32_t RealPrecision>
AAX_CLinearTaperDelegate< T, RealPrecision >::AAX_CLinearTaperDelegate (
    T minValue = 0,
    T maxValue = 1 )
```

Constructs a Linear Taper with specified minimum and maximum values.

Note

The parameter's default value should lie within the min to max range.

Parameters

in	<i>minValue</i>	
in	<i>maxValue</i>	

14.17.3 Member Function Documentation

14.17.3.1 Clone()

```
template<typename T , int32_t RealPrecision>
AAX_CLinearTaperDelegate< T, RealPrecision > * AAX_CLinearTaperDelegate< T, RealPrecision >↔
::Clone ( ) const [virtual]
```

Constructs and returns a copy of the taper delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassTaperDelegate<T>* AAX_CSubclassTaperDelegate<T>::Clone() const
{
    return new AAX_CSubclassTaperDelegate (*this);
}
```

Implements `AAX_ITaperDelegate< T >`.

14.17.3.2 GetMinimumValue()

```
template<typename T , int32_t RealPrecision = 0>
T AAX_CLinearTaperDelegate< T, RealPrecision >::GetMinimumValue ( ) const [inline], [virtual]
```

Returns the taper's minimum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.17.3.3 GetMaximumValue()

```
template<typename T , int32_t RealPrecision = 0>
T AAX_CLinearTaperDelegate< T, RealPrecision >::GetMaximumValue ( ) const [inline], [virtual]
```

Returns the taper's maximum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.17.3.4 ConstrainRealValue()

```
template<typename T , int32_t RealPrecision>
T AAX_CLinearTaperDelegate< T, RealPrecision >::ConstrainRealValue (
    T value ) const [virtual]
```

Applies a constraint to the value and returns the constrained value.

This method is useful if the taper requires a constraint beyond simple minimum and maximum real value limits.

Note

This is the function that should actually enforces the constraints in [NormalizeToReal\(\)](#) and [RealToNormalized\(\)](#).

Parameters

in	<i>value</i>	The unconstrained value
----	--------------	-------------------------

Implements [AAX_ITaperDelegate< T >](#).

14.17.3.5 NormalizedToReal()

```
template<typename T , int32_t RealPrecision>
T AAX_CLinearTaperDelegate< T, RealPrecision >::NormalizedToReal (
    double normalizedValue ) const [virtual]
```

Converts a normalized value to a real value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [RealToNormalized\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>normalizedValue</i>	The normalized value that will be converted
----	------------------------	---

Implements [AAX_ITaperDelegate< T >](#).

14.17.3.6 RealToNormalized()

```
template<typename T , int32_t RealPrecision>
double AAX\_CLinearTaperDelegate< T, RealPrecision >::RealToNormalized (
    T realValue ) const [virtual]
```

Normalizes a real parameter value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [NormalizedToReal\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>realValue</i>	The real parameter value that will be normalized
----	------------------	--

Implements [AAX_ITaperDelegate< T >](#).

14.17.3.7 Round()

```
template<typename T , int32_t RealPrecision>
T AAX\_CLinearTaperDelegate< T, RealPrecision >::Round (
    double iValue ) const [protected]
```

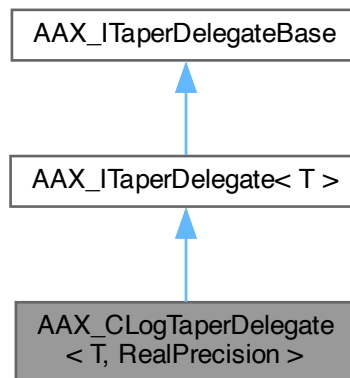
The documentation for this class was generated from the following file:

- [AAX_CLinearTaperDelegate.h](#)

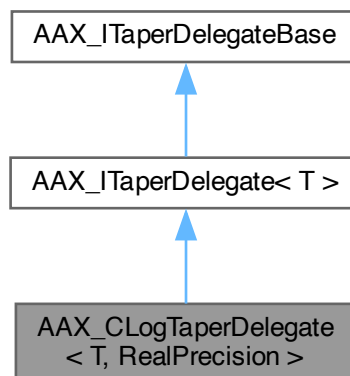
14.18 AAX_CLogTaperDelegate< T, RealPrecision > Class Template Reference

```
#include <AAX_CLogTaperDelegate.h>
```

Inheritance diagram for AAX_CLogTaperDelegate< T, RealPrecision >:



Collaboration diagram for AAX_CLogTaperDelegate< T, RealPrecision >:



14.18.1 Description

```
template<typename T, int32_t RealPrecision = 1000>  
class AAX_CLogTaperDelegate< T, RealPrecision >
```

A logarithmic taper conforming to [AAX_ITaperDelegate](#).

This taper spaces a parameter's real values between its minimum and maximum bounds, with a natural logarithmic mapping between the parameter's real and normalized values.

RealPrecision

In addition to its type templatization, this taper includes a precision template parameter. RealPrecision is a multiplier that works in conjunction with the round() function to limit the precision of the real values provided by this taper. For example, if RealPrecision is 1000, it will round to the closest 0.001 when doing any sort of value conversion. If RealPrecision is 1, it will round to the nearest integer. If RealPrecision is 1000000, it will round to the nearest 0.000001. This is particularly useful for preventing things like 1.9999999 truncating down to 1 instead of rounding up to 2.

To accomplish this behavior, the taper multiplies its unrounded parameter values by RealPrecision, rounds the result to the nearest valid value, then divides RealPrecision back out.

Rounding will be disabled if RealPrecision is set to a value less than 1

Public Member Functions

- [AAX_CLogTaperDelegate](#) (T minValue=0, T maxValue=1)
Constructs a Log Taper with specified minimum and maximum values.
- [AAX_CLogTaperDelegate](#)< T, RealPrecision > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the taper delegate.
- T [GetMinimumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's minimum real value.
- T [GetMaximumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's maximum real value.
- T [ConstrainRealValue](#) (T value) const [AAX_OVERRIDE](#)
Applies a constraint to the value and returns the constrained value.
- T [NormalizedToReal](#) (double normalizedValue) const [AAX_OVERRIDE](#)
Converts a normalized value to a real value.
- double [RealToNormalized](#) (T realValue) const [AAX_OVERRIDE](#)
Normalizes a real parameter value.

- virtual [AAX_ITaperDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the taper delegate.
- virtual T [GetMaximumValue](#) () const =0
Returns the taper's maximum real value.
- virtual T [GetMinimumValue](#) () const =0
Returns the taper's minimum real value.
- virtual T [ConstrainRealValue](#) (T value) const =0
Applies a constraint to the value and returns the constrained value.
- virtual T [NormalizedToReal](#) (double normalizedValue) const =0
Converts a normalized value to a real value.
- virtual double [RealToNormalized](#) (T realValue) const =0
Normalizes a real parameter value.

Public Member Functions inherited from [AAX_ITaperDelegateBase](#)

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

Protected Member Functions

- [T Round](#) (double iValue) const

14.18.2 Constructor & Destructor Documentation

14.18.2.1 AAX_CLogTaperDelegate()

```
template<typename T , int32_t RealPrecision>
AAX_CLogTaperDelegate< T, RealPrecision >::AAX_CLogTaperDelegate (
    T minValue = 0,
    T maxValue = 1 )
```

Constructs a Log Taper with specified minimum and maximum values.

Note

The parameter's default value should lie within the min to max range.

Parameters

in	<i>minValue</i>	
in	<i>maxValue</i>	

14.18.3 Member Function Documentation

14.18.3.1 Clone()

```
template<typename T , int32_t RealPrecision>
AAX_CLogTaperDelegate< T, RealPrecision > * AAX_CLogTaperDelegate< T, RealPrecision >::Clone
( ) const [virtual]
```

Constructs and returns a copy of the taper delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassTaperDelegate<T>* AAX_CSubclassTaperDelegate<T>::Clone() const
{
    return new AAX_CSubclassTaperDelegate(*this);
}
```

Implements [AAX_ITaperDelegate< T >](#).

14.18.3.2 GetMinimumValue()

```
template<typename T , int32_t RealPrecision = 1000>
T AAX_CLogTaperDelegate< T, RealPrecision >::GetMinimumValue ( ) const [inline], [virtual]
```

Returns the taper's minimum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.18.3.3 GetMaximumValue()

```
template<typename T , int32_t RealPrecision = 1000>
T AAX_CLogTaperDelegate< T, RealPrecision >::GetMaximumValue ( ) const [inline], [virtual]
```

Returns the taper's maximum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.18.3.4 ConstrainRealValue()

```
template<typename T , int32_t RealPrecision>
T AAX_CLogTaperDelegate< T, RealPrecision >::ConstrainRealValue (
    T value ) const [virtual]
```

Applies a constraint to the value and returns the constrained value.

This method is useful if the taper requires a constraint beyond simple minimum and maximum real value limits.

Note

This is the function that should actually enforces the constraints in [NormalizeToReal\(\)](#) and [RealToNormalized\(\)](#).

Parameters

in	value	The unconstrained value
----	-------	-------------------------

Implements [AAX_ITaperDelegate< T >](#).

14.18.3.5 NormalizedToReal()

```
template<typename T , int32_t RealPrecision>
T AAX_CLogTaperDelegate< T, RealPrecision >::NormalizedToReal (
    double normalizedValue ) const [virtual]
```

Converts a normalized value to a real value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [RealToNormalized\(\)](#), to within the roundoff precision of the individual taper implementation.

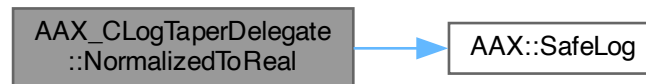
Parameters

in	<i>normalizedValue</i>	The normalized value that will be converted
----	------------------------	---

Implements [AAX_ITaperDelegate< T >](#).

References [AAX::SafeLog\(\)](#).

Here is the call graph for this function:



14.18.3.6 RealToNormalized()

```
template<typename T , int32_t RealPrecision>
double AAX\_CLogTaperDelegate< T, RealPrecision >::RealToNormalized (
    T realValue ) const [virtual]
```

Normalizes a real parameter value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [NormalizedToReal\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>realValue</i>	The real parameter value that will be normalized
----	------------------	--

Implements [AAX_ITaperDelegate< T >](#).

References [AAX::SafeLog\(\)](#).

Here is the call graph for this function:



14.18.3.7 Round()

```

template<typename T , int32_t RealPrecision>
T AAX_CLogTaperDelegate< T, RealPrecision >::Round (
    double iValue ) const [protected]
  
```

The documentation for this class was generated from the following file:

- [AAX_CLogTaperDelegate.h](#)

14.19 AAX_CMidiPacket Struct Reference

```
#include <AAX.h>
```

14.19.1 Description

Packet structure for MIDI data.

See also

[AAX_CMidiStream](#)

Legacy Porting Notes Corresponds to DirectMidiPacket in the legacy SDK

Public Attributes

- uint32_t [mTimestamp](#)
This is the playback time at which the MIDI event should occur, relative to the beginning of the current audio buffer.
- uint32_t [mLength](#)
The length of MIDI message, in terms of bytes.
- unsigned char [mData](#) [4]
The MIDI message itself. Each array element is one byte of the message, with the 0th element being the first byte.
- [AAX_CBoolean mIsImmediate](#)
Indicates that the message is to be sent as soon as possible.

14.19.2 Member Data Documentation

14.19.2.1 mTimestamp

```
uint32_t AAX_CMidiPacket::mTimestamp
```

This is the playback time at which the MIDI event should occur, relative to the beginning of the current audio buffer.

14.19.2.2 mLength

```
uint32_t AAX_CMidiPacket::mLength
```

The length of MIDI message, in terms of bytes.

14.19.2.3 mData

```
unsigned char AAX_CMidiPacket::mData[4]
```

The MIDI message itself. Each array element is one byte of the message, with the 0th element being the first byte.

Referenced by [AAX::IsAccentedClick\(\)](#), [AAX::IsAllNotesOff\(\)](#), [AAX::IsNoteOff\(\)](#), [AAX::IsNoteOn\(\)](#), and [AAX::IsUnaccentedClick\(\)](#).

14.19.2.4 mIsImmediate

```
AAX_CBoolean AAX_CMidiPacket::mIsImmediate
```

Indicates that the message is to be sent as soon as possible.

Host Compatibility Notes This value is not currently set. Use `mTimestamp == 0` to detect immediate packets

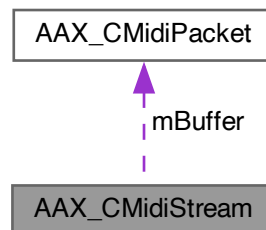
The documentation for this struct was generated from the following file:

- [AAX.h](#)

14.20 AAX_CMidiStream Struct Reference

```
#include <AAX.h>
```

Collaboration diagram for AAX_CMidiStream:



14.20.1 Description

MIDI stream data structure used by [AAX_IMIDINode](#).

For [MIDI input](#), mBufferSize is set by the AAX host when the buffer is filled.

For [MIDI output](#), the plug-in sets mBufferSize with the number of [AAX_CMidiPacket](#) objects it has filled mBuffer with. The AAX host will reset mBufferSize to 0 after it has received the buffer of MIDI.

System Exclusive (SysEx) messages that are greater than 4 bytes in length can be transmitted via a series of concurrent [AAX_CMidiPacket](#) objects in mBuffer. In accordance with the MIDI Specification, 0xF0 indicates the beginning of a SysEx message and 0xF7 indicates its end.

Legacy Porting Notes Corresponds to DirectMidiNode in the legacy SDK

Public Attributes

- `uint32_t mBufferSize`
The number of [AAX_CMidiPacket](#) objects contained in the node's buffer.
- `AAX_CMidiPacket * mBuffer`
Pointer to the first element of the node's buffer.

14.20.2 Member Data Documentation

14.20.2.1 mBufferSize

```
uint32_t AAX_CMidiStream::mBufferSize
```

The number of [AAX_CMidiPacket](#) objects contained in the node's buffer.

14.20.2.2 mBuffer

```
AAX_CMidiPacket* AAX_CMidiStream::mBuffer
```

Pointer to the first element of the node's buffer.

The documentation for this struct was generated from the following file:

- [AAX.h](#)

14.21 AAX_CMonolithicParameters Class Reference

```
#include <AAX_CMonolithicParameters.h>
```


This extension to [AAX_CEffectParameters](#) adds some conveniences for Virtual Instrument (VI) plug-ins and for other plug-ins that use a monolithic processing object, i.e. an object that combines state data with the audio render routine in a single object.

- The [RenderAudio](#) method provides a direct audio processing callback within the data model object. Perform all audio processing in this method.
- The [StaticDescribe](#) method establishes a generic MIDI processing context for the Effect. Call this method from the plug-in's [Description callback](#) implementation.
- The [AddSynchronizedParameter](#) method provides a mechanism for synchronizing parameter updates with the real-time thread, allowing deterministic, accurate automation playback. For more information about this feature, see [Fixing timing issues due to shared data](#)

Note

This convenience class assumes a monolithic processing environment (i.e. [AAX_eConstraintLocationMask_DataModel](#).) This precludes the use of [AAX_CMonolithicParameters](#) -derived Effects in distributed-processing formats such as AAX DSP.

Public Member Functions

- [AAX_CMonolithicParameters](#) (void)
- [~AAX_CMonolithicParameters](#) (void) [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_CEffectParameters](#)

- [AAX_CEffectParameters](#) (void)
- [~AAX_CEffectParameters](#) (void) [AAX_OVERRIDE](#)
- [AAX_CEffectParameters & operator=](#) (const [AAX_CEffectParameters](#) &other)
- [AAX_Result Initialize](#) ([IACFUnknown](#) *iController) [AAX_OVERRIDE](#)
Main data model initialization. Called when plug-in instance is first instantiated.
- [AAX_Result Uninitialize](#) (void) [AAX_OVERRIDE](#)
Main data model uninitialization.
- [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize) [AAX_OVERRIDE](#)
Notification Hook.
- [AAX_Result GetNumberOfParameters](#) (int32_t *oNumControls) const [AAX_OVERRIDE](#)
CALL: Retrieves the total number of plug-in parameters.
- [AAX_Result GetMasterBypassParameter](#) ([AAX_IString](#) *oIDString) const [AAX_OVERRIDE](#)
CALL: Retrieves the ID of the plug-in's Master Bypass parameter.
- [AAX_Result GetParameterIsAutomatable](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *oAutomatable) const [AAX_OVERRIDE](#)
CALL: Retrieves information about a parameter's automatable status.
- [AAX_Result GetParameterNumberOfSteps](#) ([AAX_CParamID](#) iParameterID, int32_t *oNumSteps) const [AAX_OVERRIDE](#)
CALL: Retrieves the number of discrete steps for a parameter.
- [AAX_Result GetParameterName](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName) const [AAX_OVERRIDE](#)

CALL: Retrieves the full name for a parameter.

- [AAX_Result GetParameterNameOfLength](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName, [int32_t](#) iNameLength) const [AAX_OVERRIDE](#)

CALL: Retrieves an abbreviated name for a parameter.

- [AAX_Result GetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) *oValue) const [AAX_OVERRIDE](#)

CALL: Retrieves default value of a parameter.

- [AAX_Result SetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue) [AAX_OVERRIDE](#)

CALL: Sets the default value of a parameter.

- [AAX_Result GetParameterType](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterType](#) *oParameterType) const [AAX_OVERRIDE](#)

CALL: Retrieves the type of a parameter.

- [AAX_Result GetParameterOrientation](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterOrientation](#) *oParameterOrientation) const [AAX_OVERRIDE](#)

CALL: Retrieves the orientation that should be applied to a parameter's controls.

- [AAX_Result GetParameter](#) ([AAX_CParamID](#) iParameterID, [AAX_IParameter](#) **oParameter) [AAX_OVERRIDE](#)

CALL: Retrieves an arbitrary setting within a parameter.

- [AAX_Result GetParameterIndex](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) *oControllIndex) const [AAX_OVERRIDE](#)

CALL: Retrieves the index of a parameter.

- [AAX_Result GetParameterIDFromIndex](#) ([int32_t](#) iControllIndex, [AAX_IString](#) *oParameterIDString) const [AAX_OVERRIDE](#)

CALL: Retrieves the ID of a parameter.

- [AAX_Result GetParameterValueInfo](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) iSelector, [int32_t](#) *oValue) const [AAX_OVERRIDE](#)

CALL: Retrieves a property of a parameter.

- [AAX_Result GetParameterValueFromString](#) ([AAX_CParamID](#) iParameterID, [double](#) *oValue, const [AAX_IString](#) &iValueString) const [AAX_OVERRIDE](#)

CALL: Converts a value string to a value.

- [AAX_Result GetParameterStringFromValue](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const [AAX_OVERRIDE](#)

CALL: Converts a normalized parameter value into a string representing its corresponding real value.

- [AAX_Result GetParameterValueString](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const [AAX_OVERRIDE](#)

CALL: Retrieves the value string associated with a parameter's current value.

- [AAX_Result GetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) *oValuePtr) const [AAX_OVERRIDE](#)

CALL: Retrieves a parameter's current value.

- [AAX_Result SetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue) [AAX_OVERRIDE](#)

CALL: Sets the specified parameter to a new value.

- [AAX_Result SetParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue) [AAX_OVERRIDE](#)

CALL: Sets the specified parameter to a new value relative to its current value.

- [AAX_Result TouchParameter](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)

"Touches" (locks) a parameter in the automation system to a particular control in preparation for updates

- [AAX_Result ReleaseParameter](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)

Releases a parameter from a "touched" state.

- [AAX_Result UpdateParameterTouch](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iTouchState) [AAX_OVERRIDE](#)

Sets a "touched" state on a parameter.

- [AAX_Result UpdateParameterNormalizedRelative](#) (AAX_CParamID iParameterID, double iValue) [AAX_OVERRIDE](#)
Updates a single parameter's state to its current value, as a difference with the parameter's previous value.

- [AAX_Result GetNumberOfChunks](#) (int32_t *oNumChunks) const [AAX_OVERRIDE](#)
Retrieves the number of chunks used by this plug-in.
- [AAX_Result GetChunkIDFromIndex](#) (int32_t iIndex, AAX_CTypeID *oChunkID) const [AAX_OVERRIDE](#)
Retrieves the ID associated with a chunk index.
- [AAX_Result GetChunkSize](#) (AAX_CTypeID iChunkID, uint32_t *oSize) const [AAX_OVERRIDE](#)
Get the size of the data structure that can hold all of a chunk's information.
- [AAX_Result GetChunk](#) (AAX_CTypeID iChunkID, AAX_SPlugInChunk *oChunk) const [AAX_OVERRIDE](#)
Fills a block of data with chunk information representing the plug-in's current state.
- [AAX_Result SetChunk](#) (AAX_CTypeID iChunkID, const AAX_SPlugInChunk *iChunk) [AAX_OVERRIDE](#)
Restores a set of plug-in parameters based on chunk information.
- [AAX_Result CompareActiveChunk](#) (const AAX_SPlugInChunk *iChunkP, AAX_CBoolean *oIsEqual) const [AAX_OVERRIDE](#)
Determine if a chunk represents settings that are equivalent to the plug-in's current state.
- [AAX_Result GetNumberOfChanges](#) (int32_t *oNumChanges) const [AAX_OVERRIDE](#)
Retrieves the number of parameter changes made since the plug-in's creation.

- [AAX_Result GetCurveData](#) (AAX_CTypeID iCurveType, const float *iValues, uint32_t iNumValues, float *o↔Values) const [AAX_OVERRIDE](#)
Generate a set of output values based on a set of given input values.
- [AAX_Result GetCurveDataMeterIds](#) (AAX_CTypeID iCurveType, uint32_t *oXMeterId, uint32_t *oYMeterId) const [AAX_OVERRIDE](#)
Indicates which meters correspond to the X and Y axes of the EQ or Dynamics graph.
- [AAX_Result GetCurveDataDisplayRange](#) (AAX_CTypeID iCurveType, float *oXMin, float *oXMax, float *o↔YMin, float *oYMax) const [AAX_OVERRIDE](#)
Determines the range of the graph shown by the plug-in.
- [AAX_Result UpdatePageTable](#) (uint32_t inTableType, int32_t inTablePageSize, IACFUnknown *iHost↔Unknown, IACFUnknown *ioPageTableUnknown) const [AAX_OVERRIDE](#) [AAX_FINAL](#)
Allow the plug-in to update its page tables.

- [AAX_Result GetCustomData](#) (AAX_CTypeID iDataBlockID, uint32_t inDataSize, void *oData, uint32_t *o↔DataWritten) const [AAX_OVERRIDE](#)
An optional interface hook for getting custom data from another module.
- [AAX_Result SetCustomData](#) (AAX_CTypeID iDataBlockID, uint32_t inDataSize, const void *iData) [AAX_OVERRIDE](#)
An optional interface hook for setting custom data for use by another module.

- [AAX_Result DoMIDITransfers](#) () [AAX_OVERRIDE](#)
MIDI update callback.
- [AAX_Result UpdateMIDINodes](#) (AAX_CFieldIndex inFieldIndex, AAX_CMidiPacket &iPacket) [AAX_OVERRIDE](#)
MIDI update callback.
- [AAX_Result UpdateControlMIDINodes](#) (AAX_CTypeID nodeID, AAX_CMidiPacket &iPacket) [AAX_OVERRIDE](#)
MIDI update callback for control MIDI nodes.

- [AAX_Result RenderAudio_Hybrid](#) (AAX_SHybridRenderInfo *ioRenderInfo) [AAX_OVERRIDE](#)
Hybrid audio render function.

- [AAX_IController](#) * [Controller](#) ()
Access to the Effect controller.
- const [AAX_IController](#) * [Controller](#) () const
const access to the Effect controller
- [AAX_ITransport](#) * [Transport](#) ()
Access to the Transport object.
- const [AAX_ITransport](#) * [Transport](#) () const
const access to the Transport object
- [AAX_IAutomationDelegate](#) * [AutomationDelegate](#) ()
Access to the Effect's automation delegate.
- const [AAX_IAutomationDelegate](#) * [AutomationDelegate](#) () const
const access to the Effect's automation delegate

Public Member Functions inherited from [AAX_IEffectParameters](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid
- [AAX_DELETE](#) ([AAX_IEffectParameters](#) &operator=(const [AAX_IEffectParameters](#) &))

Auxiliary UI methods

Auxiliary UI methods

Hybrid audio methods

MIDI methods

Initialization and uninitialization

AAX host and plug-in event notification

Parameter information

These methods are used by the AAX host to retrieve information about the plug-in's data model.

For information about adding parameters to the plug-in and otherwise modifying the plug-in's data model, see [AAX_CParameterManager](#). For information about parameters, see [AAX_IParameter](#).

Parameter setters and getters

These methods are used by the AAX host and by the plug-in's UI to retrieve and modify the values of the plug-in's parameters.

Note

The parameter setters in this section may generate asynchronous requests.

Automated parameter helpers

These methods are used to lock and unlock automation system 'resources' when updating automatable parameters.

Note

You should never need to override these methods to extend their behavior beyond what is provided in [AAX_CEffectParameters](#) and [AAX_IParameter](#)

Asynchronous parameter update methods

These methods are called by the AAX host when parameter values have been updated. They are called by the host and can be triggered by other plug-in modules via calls to [AAX_IParameter](#)'s `SetValue` methods, e.g. [SetValueWithFloat\(\)](#)

These methods are responsible for updating parameter values.

Do not call these methods directly! To ensure proper synchronization and to avoid problematic dependency chains, other methods (e.g. [SetParameterNormalizedValue\(\)](#)) and components (e.g. [AAX_IEffectGUI](#)) should always call a `SetValue` method on [AAX_IParameter](#) to update parameter values. The `SetValue` method will properly manage automation locks and other system resources.

State reset handlers**Chunk methods**

These methods are used to save and restore collections of plug-in state information, known as chunks. Chunks are used by the host when saving or restoring presets and session settings and when providing "compare" functionality for plug-ins.

The default implementation of these methods in [AAX_CEffectParameters](#) supports a single chunk that includes state information for all of the plug-in's registered parameters. Override all of these methods to add support for additional chunks in your plug-in, for example if your plug-in contains any persistent state that is not encapsulated by its set of registered parameters.

Warning

Remember that plug-in chunk data may be loaded on a different platform from the one where it is saved. All data structures in the chunk must be properly data-aligned for compatibility across all platforms that the plug-in supports. See [AAX_ALIGN_FILE_ALG](#) for notes about common cross-platform pitfalls for data structure alignment.

For reference, see also:

- [AAX_CChunkDataParser](#)
- [AAX_SPlugInChunk](#)

Thread methods**Auxiliary UI methods****Custom data methods**

These functions exist as a proxiable way to move data between different modules (e.g. [AAX_IEffectParameters](#) and [AAX_IEffectGUI](#).) Using these, the GUI can query any data through [GetCustomData\(\)](#) with a plug-in defined `typeID`, `void` and size. This has an advantage over just sharing memory in that this function can work as a remote proxy as we enable those sorts of features later in the platform. Likewise, the GUI can also set arbitrary data on the data model by using the [SetCustomData\(\)](#) function with the same idea.*

Note

These are plug-in internal only. They are not called from the host right now, or likely ever.

MIDI methods

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Protected Types

- typedef std::pair< [AAX_CParamID](#) const, const [AAX_IParameterValue](#) * > [TParamValPair](#)

Protected Member Functions

Real-time functions

Virtual functions called on the real-time thread

- virtual void [RenderAudio](#) ([AAX_SInstrumentRenderInfo](#) *ioRenderInfo, const [TParamValPair](#) *in← SynchronizedParamValues[], int32_t inNumSynchronizedParamValues)

Configuration methods

- void [AddSynchronizedParameter](#) (const [AAX_IParameter](#) &inParameter)

Protected Member Functions inherited from [AAX_CEffectParameters](#)

- [AAX_Result](#) [SetTaperDelegate](#) ([AAX_CParamID](#) iParameterID, [AAX_ITaperDelegateBase](#) &iTaperDelegate, bool iPreserveValue)
- [AAX_Result](#) [SetDisplayDelegate](#) ([AAX_CParamID](#) iParameterID, [AAX_IDisplayDelegateBase](#) &iDisplay← Delegate)
- bool [IsParameterTouched](#) ([AAX_CParamID](#) iParameterID) const
- bool [IsParameterLinkReady](#) ([AAX_CParamID](#) inParameterID, [AAX_EUpdateSource](#) inSource) const
- virtual [AAX_Result](#) [EffectInit](#) (void)
Initialization helper routine. Called from [AAX_CEffectParameters::Initialize](#).
- virtual [AAX_Result](#) [UpdatePageTable](#) (uint32_t, int32_t, [AAX_IPageTable](#) &) const
- void [FilterParameterIDOnSave](#) ([AAX_CParamID](#) controlId)
CALL: Indicates the indices of parameters that should not be saved in the default [AAX_CEffectParameters](#) chunk.
- void [BuildChunkData](#) (void) const
Clears out the current chunk in Chunk Parser and adds all of the new values. Used by default implementations of [GetChunk\(\)](#) and [GetChunkSize\(\)](#).

Convenience Layer Methods

Note

You should not need to override these methods, but if you do, make sure to call into the base class.

- [AAX_Result UpdateParameterNormalizedValue](#) ([AAX_CParamID](#) iParamID, double aValue, [AAX_EUpdateSource](#) inSource) [AAX_OVERRIDE](#)
Updates a single parameter's state to its current value.
- [AAX_Result GenerateCoefficients](#) () [AAX_OVERRIDE](#)
Generates and dispatches new coefficient packets.
- [AAX_Result ResetFieldData](#) ([AAX_CFieldIndex](#) iFieldIndex, void *oData, uint32_t iDataSize) const [AAX_OVERRIDE](#)
Called by the host to reset a private data field in the plug-in's algorithm.
- [AAX_Result TimerWakeup](#) () [AAX_OVERRIDE](#)
Periodic wakeup callback for idle-time operations.
- static [AAX_Result StaticDescribe](#) ([AAX_IEffectDescriptor](#) *ioDescriptor, const [AAX_SInstrumentSetupInfo](#) &setupInfo)
- static void [AAX_CALLBACK StaticRenderAudio](#) ([AAX_SInstrumentRenderInfo](#) *const inInstancesBegin[], const void *inInstancesEnd)

Additional Inherited Members

Public Attributes inherited from [AAX_IEffectParameters](#)

- void **ppvObjOut [override](#)

Protected Attributes inherited from [AAX_CEffectParameters](#)

- int32_t mNumPlugInChanges
- int32_t mChunkSize
- [AAX_CChunkDataParser](#) mChunkParser
- int32_t mNumChunkedParameters
- [AAX_CPacketDispatcher](#) mPacketDispatcher
- [AAX_CParameterManager](#) mParameterManager
- std::set< std::string > mFilteredParameters

14.21.2 Member Typedef Documentation

14.21.2.1 TParamValPair

```
typedef std::pair<AAX_CParamID const, const AAX_IParameterValue*> AAX_CMonolithicParameters::TParamValPair
[protected]
```

14.21.3 Constructor & Destructor Documentation

14.21.3.1 AAX_CMonolithicParameters()

```
AAX_CMonolithicParameters::AAX_CMonolithicParameters (
    void )
```

14.21.3.2 ~AAX_CMonolithicParameters()

```
AAX_CMonolithicParameters::~~AAX_CMonolithicParameters (
    void )
```

14.21.4 Member Function Documentation

14.21.4.1 RenderAudio()

```
virtual void AAX_CMonolithicParameters::RenderAudio (
    AAX_SInstrumentRenderInfo * ioRenderInfo,
    const TParamValPair * inSynchronizedParamValues[],
    int32_t inNumSynchronizedParamValues ) [inline], [protected], [virtual]
```

Perform audio render

Parameters

in, out	<i>ioRenderInfo</i>	State data for the current render buffer
in	<i>inSynchronizedParamValues</i>	The parameter values which should be applied for the current render buffer

See also

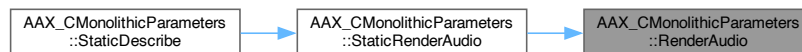
[AddSynchronizedParameter](#)

Parameters

in	<i>inNumSynchronizedParamValues</i>	The number of parameter values provided in <i>inSynchronizedParamValues</i>
----	-------------------------------------	---

Referenced by [StaticRenderAudio\(\)](#).

Here is the caller graph for this function:



14.21.4.2 AddSynchronizedParameter()

```
void AAX_CMonolithicParameters::AddSynchronizedParameter (
    const AAX_IParameter & inParameter ) [protected]
```

Add a parameter for state synchronization

A parameter should be added for synchronization if:

- It is important for the parameter's automation to be applied at the correct point on the timeline
- It is possible to quickly update the plug-in's state to reflect a parameter change

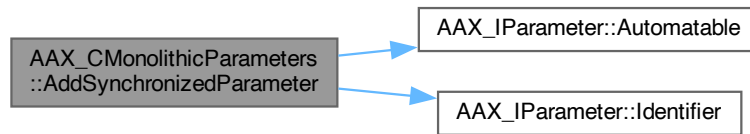
See [Parameter update timing](#) for more information

Parameters

in	<i>inParameter</i>	The parameter to be synchronized. This string will be copied internally and is not required to persist
----	--------------------	--

References [AAX_ASSERT](#), [AAX_IParameter::Automatable\(\)](#), [AAX_IParameter::Identifier\(\)](#), and [kSynchronizedParameterQueueSize](#).

Here is the call graph for this function:



14.21.4.3 UpdateParameterNormalizedValue()

```

AAX_Result AAX_CMonolithicParameters::UpdateParameterNormalizedValue (
    AAX_CParamID iParameterID,
    double iValue,
    AAX_EUpdateSource iSource ) [virtual]
  
```

Updates a single parameter's state to its current value.

Note

Do *not* call this method from the plug-in. This method should be called by the host only. To set parameter values from within the plug-in, use the [AAX_IPParameter](#) interface.

Todo FLAGGED FOR CONSIDERATION OF REVISION

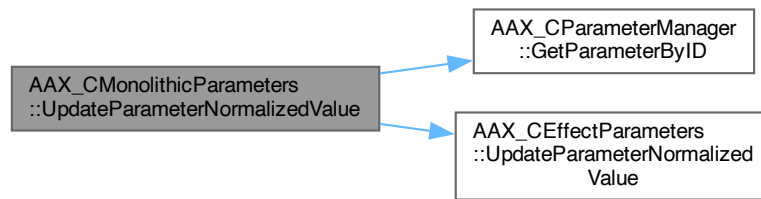
Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being updated
in	<i>iValue</i>	The parameter's current value, to which its internal state must be updated
in	<i>iSource</i>	The source of the update

Reimplemented from [AAX_CEffectParameters](#).

References [AAX_SUCCESS](#), [AAX_CParameterManager::GetParameterByID\(\)](#), [AAX_CEffectParameters::mParameterManager](#), and [AAX_CEffectParameters::UpdateParameterNormalizedValue\(\)](#).

Here is the call graph for this function:



14.21.4.4 GenerateCoefficients()

`AAX_Result AAX_CMonolithicParameters::GenerateCoefficients () [virtual]`

Generates and dispatches new coefficient packets.

This method is responsible for updating the coefficient packets associated with all parameters whose states have changed since the last call to [GenerateCoefficients\(\)](#). The host may call this method once for every parameter update, or it may "batch" parameter updates such that changes for several parameters are all handled by a single call to [GenerateCoefficients\(\)](#).

For more information on tracking parameters' statuses using the [AAX_CPacketDispatcher](#), helper class, see [AAX_CPacketDispatcher::SetDirty\(\)](#).

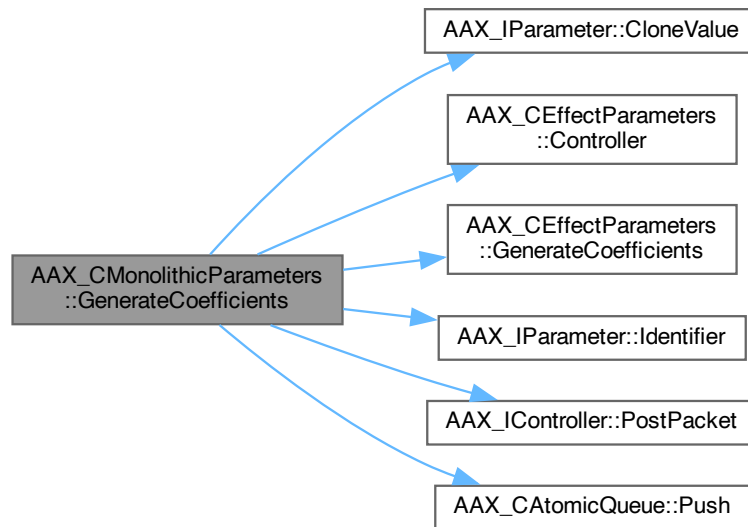
Note

Do *not* call this method from the plug-in. This method should be called by the host only. To set parameter values from within the plug-in, use the [AAX_IParameter](#) interface.

Reimplemented from [AAX_CEffectParameters](#).

References [AAX_ASSERT](#), [AAX_FIELD_INDEX](#), [AAX_SUCCESS](#), [AAX_IParameter::CloneValue\(\)](#), [AAX_CEffectParameters::Control](#), [AAX_IContainer::eStatus_Success](#), [AAX_CEffectParameters::GenerateCoefficients\(\)](#), [AAX_IParameter::Identifier\(\)](#), [AAX_IController::PostPacket\(\)](#), and [AAX_CAtomicQueue< T, S >::Push\(\)](#).

Here is the call graph for this function:



14.21.4.5 ResetFieldData()

```

AAX_Result AAX_CMonolithicParameters::ResetFieldData (
    AAX_CFieldIndex inFieldIndex,
    void * oData,
    uint32_t inDataSize ) const [virtual]
  
```

Called by the host to reset a private data field in the plug-in's algorithm.

This method is called sequentially for all private data fields on Effect initialization and during any "reset" event, such as priming for a non-real-time render. This method is called before the algorithm's optional initialization callback, and the initialized private data will be available within that callback via its context block.

See also

[Algorithm initialization.](#)

Warning

Any data structures that will be passed between platforms (for example, sent to a TI DSP in an AAX DSP plug-in) must be properly data-aligned for compatibility across both platforms. See [AAX_ALIGN_FILE_ALG](#) for more information about guaranteeing cross-platform compatibility of data structures used for algorithm processing.

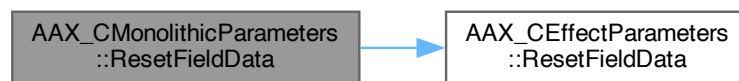
Parameters

in	<i>inFieldIndex</i>	The index of the field that is being initialized
out	<i>oData</i>	The pre-allocated block of data that should be initialized
in	<i>inDataSize</i>	The size of the data block, in bytes

Reimplemented from [AAX_CEffectParameters](#).

References [AAX_ASSERT](#), [AAX_FIELD_INDEX](#), [AAX_SUCCESS](#), [AAX_SInstrumentPrivateData::mMonolithicParametersPtr](#), and [AAX_CEffectParameters::ResetFieldData\(\)](#).

Here is the call graph for this function:



14.21.4.6 TimerWakeup()

```
AAX_Result AAX_CMonolithicParameters::TimerWakeup ( ) [virtual]
```

Periodic wakeup callback for idle-time operations.

This method is called from the host using a non-main thread. In general, it should be driven at approximately one call per 30 ms. However, the wakeup is not guaranteed to be called at any regular interval - for example, it could be held off by a high real-time processing load - and there is no host contract regarding maximum latency between wakeup calls.

This wakeup thread runs continuously and cannot be armed/disarmed or by the plug-in.

Reimplemented from [AAX_CEffectParameters](#).

References [AAX_CEffectParameters::TimerWakeup\(\)](#).

Here is the call graph for this function:



14.21.4.7 StaticDescribe()

```
AAX_Result AAX_CMonolithicParameters::StaticDescribe (
    AAX_IEffectDescriptor * ioDescriptor,
    const AAX_SInstrumentSetupInfo & setupInfo ) [static]
```

Static description callback

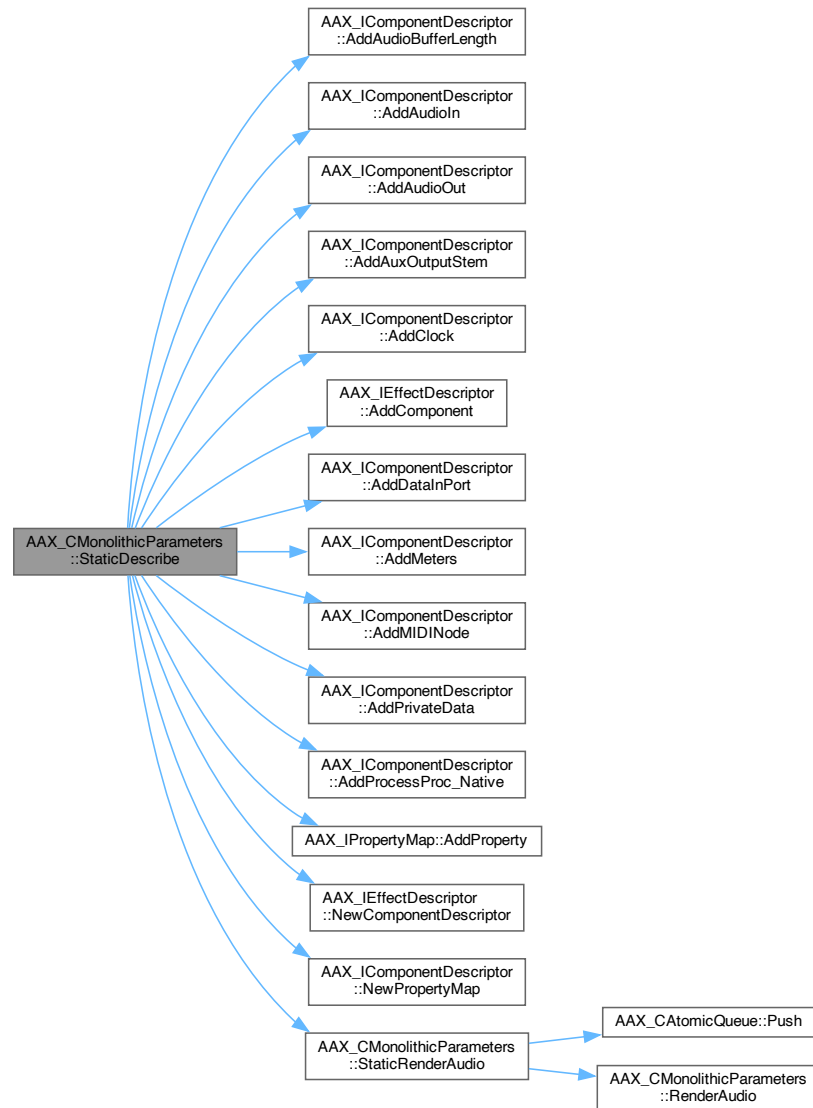
This method performs all of the basic context setup and pointer passing work

Parameters

in, out	<i>ioDescriptor</i>	
in	<i>setupInfo</i>	

References [AAX_ASSERT](#), [AAX_eConstraintLocationMask_DataModel](#), [AAX_eMIDINodeType_Global](#), [AAX_eMIDINodeType_Local](#), [AAX_eMIDINodeType_Transport](#), [AAX_ePrivateDataOptions_DefaultOptions](#), [AAX_eProperty_CanBypass](#), [AAX_eProperty_Constraint_Location](#), [AAX_eProperty_Constraint_MultiMonoSupport](#), [AAX_eProperty_HybridInputStemFormat](#), [AAX_eProperty_HybridOutputStemFormat](#), [AAX_eProperty_InputStemFormat](#), [AAX_eProperty_ManufacturerID](#), [AAX_eProperty_OutputStemFormat](#), [AAX_eProperty_PluginID_AudioSuite](#), [AAX_eProperty_PluginID_RTAS](#), [AAX_eProperty_ProductID](#), [AAX_eProperty_UsesClientGUI](#), [AAX_eProperty_UsesTransport](#), [AAX_ERROR_NULL_OBJECT](#), [AAX_eStemFormat_None](#), [AAX_FIELD_INDEX](#), [AAX_IComponentDescriptor::AddAudioBufferLength\(\)](#), [AAX_IComponentDescriptor::AddAudioOut\(\)](#), [AAX_IComponentDescriptor::AddAuxOutputStem\(\)](#), [AAX_IComponentDescriptor::AddComponent\(\)](#), [AAX_IComponentDescriptor::AddDataInPort\(\)](#), [AAX_IComponentDescriptor::AddMeters\(\)](#), [AAX_IComponentDescriptor::AddMIDIChannelMask\(\)](#), [AAX_IComponentDescriptor::AddMIDINode\(\)](#), [AAX_IComponentDescriptor::AddPrivateData\(\)](#), [AAX_IComponentDescriptor::AddPropertyMap\(\)](#), [AAX_IPropertyMap::AddProperty\(\)](#), [kMaxAdditionalMIDINodes](#), [kMaxAuxOutputStems](#), [AAX_SInstrumentSetupInfo::mAudioSuiteID](#), [AAX_SInstrumentSetupInfo::mAuxOutputStemFormats](#), [AAX_SInstrumentSetupInfo::mAuxOutputStemNames](#), [AAX_SInstrumentSetupInfo::mCanBypass](#), [AAX_SInstrumentSetupInfo::mGlobalMIDIEventMask](#), [AAX_SInstrumentSetupInfo::mGlobalMIDIEventMask](#), [AAX_SInstrumentSetupInfo::mHybridInputStemFormat](#), [AAX_SInstrumentSetupInfo::mHybridOutputStemFormat](#), [AAX_SInstrumentSetupInfo::mInputMIDIChannelMask](#), [AAX_SInstrumentSetupInfo::mInputMIDINodeName](#), [AAX_SInstrumentSetupInfo::mInputStemFormat](#), [AAX_SInstrumentSetupInfo::mManufacturerID](#), [AAX_SInstrumentSetupInfo::mMeterID](#), [AAX_SInstrumentSetupInfo::mMultiMonoSupport](#), [AAX_SInstrumentSetupInfo::mNeedsGlobalMIDI](#), [AAX_SInstrumentSetupInfo::mNeedsTransport](#), [AAX_SInstrumentSetupInfo::mNumAdditionalInputMIDINodes](#), [AAX_SInstrumentSetupInfo::mNumAuxOutputStems](#), [AAX_SInstrumentSetupInfo::mNumMeters](#), [AAX_SInstrumentSetupInfo::mOutputStemFormat](#), [AAX_SInstrumentSetupInfo::mPluginID](#), [AAX_SInstrumentSetupInfo::mProductID](#), [AAX_SInstrumentSetupInfo::mUseHostGeneratedData](#), [AAX_IEffectDescriptor::NewComponentDescriptor\(\)](#), [AAX_IComponentDescriptor::NewPropertyMap\(\)](#), and [StaticRenderAudio\(\)](#).

Here is the call graph for this function:



14.21.4.8 StaticRenderAudio()

```
void AAX_CALLBACK AAX_CMonolithicParameters::StaticRenderAudio (
    AAX_SInstrumentRenderInfo *const inInstancesBegin[],
    const void * inInstancesEnd ) [static]
```

Static RenderAudio (Called by the host)

Plug-ins should override `AAX_CMonolithicParameters::RenderAudio()`

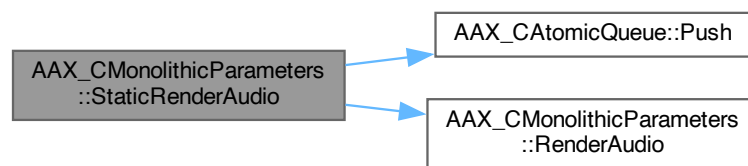
Parameters

in	<i>inInstancesBegin</i>	
in	<i>inInstancesEnd</i>	

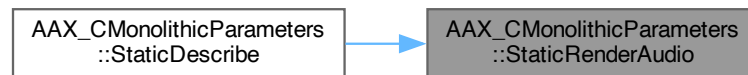
References [AAX_ASSERT](#), [AAX_IContainer::eStatus_Success](#), [AAX_SInstrumentPrivateData::mMonolithicParametersPtr](#), [AAX_CAtomicQueue< T, S >::Push\(\)](#), and [RenderAudio\(\)](#).

Referenced by [StaticDescribe\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [AAX_CMonolithicParameters.h](#)
- [AAX_CMonolithicParameters.cpp](#)

14.22 AAX_CMutex Class Reference

```
#include <AAX_CMutex.h>
```

14.22.1 Description

Mutex with try lock functionality.

Public Member Functions

- [AAX_CMutex](#) ()
- [~AAX_CMutex](#) ()
- bool [Lock](#) ()
- void [Unlock](#) ()
- bool [Try_Lock](#) ()

14.22.2 Constructor & Destructor Documentation

14.22.2.1 AAX_CMutex()

```
AAX_CMutex::AAX_CMutex ( )
```

14.22.2.2 ~AAX_CMutex()

```
AAX_CMutex::~~AAX_CMutex ( )
```

14.22.3 Member Function Documentation

14.22.3.1 Lock()

```
bool AAX_CMutex::Lock ( )
```

Referenced by [AAX_StLock_Guard::AAX_StLock_Guard\(\)](#).

Here is the caller graph for this function:

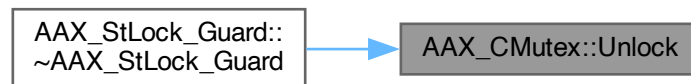


14.22.3.2 Unlock()

```
void AAX_CMutex::Unlock ( )
```

Referenced by [AAX_StLock_Guard::~~AAX_StLock_Guard\(\)](#).

Here is the caller graph for this function:



14.22.3.3 Try_Lock()

```
bool AAX_CMutex::Try_Lock ( )
```

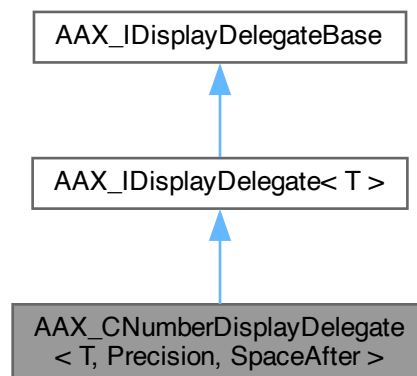
The documentation for this class was generated from the following file:

- [AAX_CMutex.h](#)

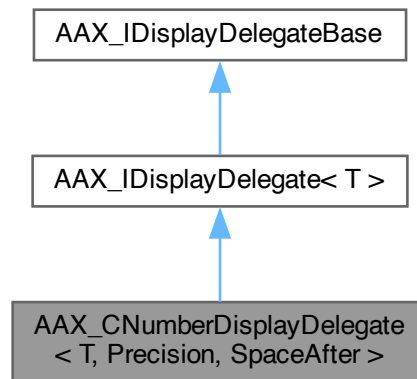
14.23 AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter > Class Template Reference

```
#include <AAX_CNumberDisplayDelegate.h>
```

Inheritance diagram for `AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >`:



Collaboration diagram for AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >:



14.23.1 Description

```
template<typename T, uint32_t Precision = 2, uint32_t SpaceAfter = 0>
class AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >
```

A numeric display format conforming to [AAX_IDisplayDelegate](#).

This display delegate converts a parameter value to a numeric string using a specified precision.

Public Member Functions

- [AAX_CNumberDisplayDelegate](#) * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.23.2 Member Function Documentation

14.23.2.1 Clone()

```
template<typename T , uint32_t Precision, uint32_t SpaceAfter>
AAX\_CNumberDisplayDelegate< T, Precision, SpaceAfter > * AAX\_CNumberDisplayDelegate< T, Precision,
SpaceAfter >::Clone ( ) const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>*    AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.23.2.2 ValueToString() [1/2]

```
template<typename T , uint32_t Precision, uint32_t SpaceAfter>
bool AAX\_CNumberDisplayDelegate< T, Precision, SpaceAfter >::ValueToString (
    T value,
    AAX\_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to value

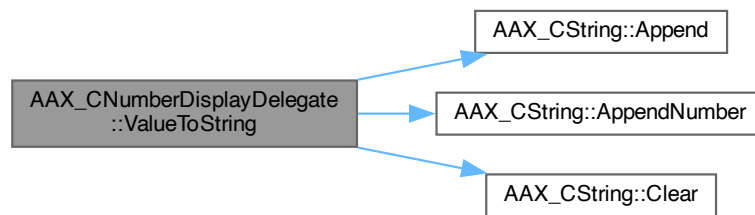
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Append\(\)](#), [AAX_CString::AppendNumber\(\)](#), and [AAX_CString::Clear\(\)](#).

Here is the call graph for this function:



14.23.2.3 ValueToString() [2/2]

```

template<typename T , uint32_t Precision, uint32_t SpaceAfter>
bool AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
  
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

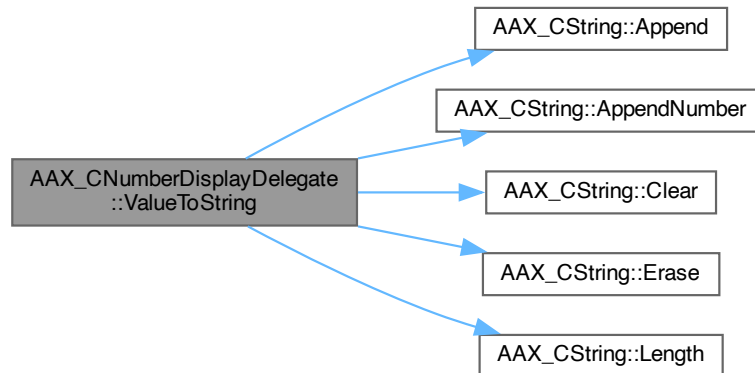
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Append\(\)](#), [AAX_CString::AppendNumber\(\)](#), [AAX_CString::Clear\(\)](#), [AAX_CString::Erase\(\)](#), and [AAX_CString::Length\(\)](#).

Here is the call graph for this function:



14.23.2.4 StringToValue()

```

template<typename T , uint32_t Precision, uint32_t SpaceAfter>
bool AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
  
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

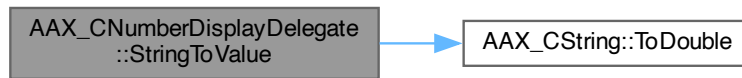
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::ToDouble\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [AAX_CNumberDisplayDelegate.h](#)

14.24 AAX_Component< aContextType > Class Template Reference

```
#include <AAX_Callbacks.h>
```

14.24.1 Description

```
template<typename aContextType>
class AAX_Component< aContextType >
```

Empty class containing type declarations for the AAX algorithm and associated callbacks.

Public Types

- typedef void([AAX_CALLBACK](#) * [CProcessProc](#)) (aContextType *const inContextPtrsBegin[], const void *inContextPtrsEnd)
- typedef void *([AAX_CALLBACK](#) * [CPacketAllocator](#)) (const aContextType *inContextPtr, [AAX_CFieldIndex](#) inOutputPort, [AAX_CTimestamp](#) inTimestamp)
- typedef int32_t([AAX_CALLBACK](#) * [CInstanceInitProc](#)) (const aContextType *inInstanceContextPtr, [AAX_EComponentInstanceInitAction](#) iAction)
- typedef int32_t([AAX_CALLBACK](#) * [CBackgroundProc](#)) (void)
- typedef void([AAX_CALLBACK](#) * [CInitPrivateDataProc](#)) ([AAX_CFieldIndex](#) inFieldIndex, void *inNewBlock, int32_t inSize, [IACFUnknown](#) *const inController)

14.24.2 Member Typedef Documentation

14.24.2.1 CProcessProc

```
template<typename aContextType >
typedef void(AAX_CALLBACK * AAX_Component< aContextType >::CProcessProc) (aContextType *const
inContextPtrsBegin[], const void *inContextPtrsEnd)
```

14.24.2.2 CPacketAllocator

```
template<typename aContextType >
typedef void *(AAX_CALLBACK * AAX_Component< aContextType >::CPacketAllocator) (const a↵
ContextType *inContextPtr, AAX_CFieldIndex inOutputPort, AAX_CTimestamp inTimestamp)
```

14.24.2.3 CInstanceInitProc

```
template<typename aContextType >
typedef int32_t(AAX_CALLBACK * AAX_Component< aContextType >::CInstanceInitProc) (const a↵
ContextType *inInstanceContextPtr, AAX_EComponentInstanceInitAction iAction)
```

14.24.2.4 CBackgroundProc

```
template<typename aContextType >
typedef int32_t(AAX_CALLBACK * AAX_Component< aContextType >::CBackgroundProc) (void)
```

14.24.2.5 CInitPrivateDataProc

```
template<typename aContextType >
typedef void(AAX_CALLBACK * AAX_Component< aContextType >::CInitPrivateDataProc) (AAX_CFieldIndex
inFieldIndex, void *inNewBlock, int32_t inSize, IACFUnknown *const inController)
```

The documentation for this class was generated from the following file:

- [AAX_Callbacks.h](#)

14.25 AAX_CPacket Class Reference

```
#include <AAX_CPacketDispatcher.h>
```


14.25.1 Description

Container for packet-related data.

This class collects a number of packet-related data into the same object and provides a facility for tracking when the parameter is "dirty", i.e. after its value has been updated and before an associated packet has not been posted.

Public Member Functions

- [AAX_CPacket](#) ([AAX_CFieldIndex](#) inFieldIndex)
- [~AAX_CPacket](#) ()
- `template<typename DataType >`
`DataType * GetPtr ()`
- `void SetDirty (bool iDirty)`
- `bool IsDirty () const`
- `AAX_CFieldIndex GetID () const`
- `uint32_t GetSize () const`
- `template<> const void * GetPtr ()`

14.25.2 Constructor & Destructor Documentation

14.25.2.1 AAX_CPacket()

```
AAX_CPacket::AAX_CPacket (
    AAX\_CFieldIndex inFieldIndex ) [inline]
```

14.25.2.2 ~AAX_CPacket()

```
AAX_CPacket::~~AAX_CPacket ( ) [inline]
```

14.25.3 Member Function Documentation

14.25.3.1 GetPtr() [1/2]

```
template<typename DataType >
DataType * AAX_CPacket::GetPtr ( ) [inline]
```

14.25.3.2 SetDirty()

```
void AAX_CPacket::SetDirty (
    bool iDirty ) [inline]
```

14.25.3.3 IsDirty()

```
bool AAX_CPacket::IsDirty ( ) const [inline]
```

14.25.3.4 GetID()

```
AAX\_CFieldIndex AAX_CPacket::GetID ( ) const [inline]
```

14.25.3.5 GetSize()

```
uint32_t AAX_CPacket::GetSize ( ) const [inline]
```

14.25.3.6 GetPtr() [2/2]

```
template<>
const void * AAX_CPacket::GetPtr ( ) [inline]
```

The documentation for this class was generated from the following file:

- [AAX_CPacketDispatcher.h](#)

14.26 AAX_CPacketDispatcher Class Reference

```
#include <AAX_CPacketDispatcher.h>
```

14.26.1 Description

Helper class for managing AAX packet posting.

This optional class can be used to associate individual parameters with custom update callbacks. The update callbacks for all "dirty" parameters are triggered whenever [AAX_CPacketDispatcher::Dispatch\(\)](#) is called. The resulting coefficient data is then posted to the [AAX_IController](#) automatically by the packet dispatcher.

The packet dispatcher supports many-to-one relationships between parameters and handler callbacks, so a single callback may be registered for several related parameters.

See also

[AAX_CEffectParameters::EffectInit\(\)](#)

Public Member Functions

- [AAX_CPacketDispatcher](#) ()
- [~AAX_CPacketDispatcher](#) ()
- void [Initialize](#) ([AAX_IController](#) *iPlugIn, [AAX_IEffectParameters](#) *iEffectParameters)
- [AAX_Result](#) [RegisterPacket](#) ([AAX_CParamID](#) paramID, [AAX_CFieldIndex](#) portID, const [AAX_IPacketHandler](#) *iHandler)
- template<class TWorker, typename Func >
[AAX_Result](#) [RegisterPacket](#) ([AAX_CParamID](#) paramID, [AAX_CFieldIndex](#) portID, TWorker *iPt2Object, Func infPt)
- [AAX_Result](#) [RegisterPacket](#) ([AAX_CParamID](#) paramID, [AAX_CFieldIndex](#) portID)
- [AAX_Result](#) [SetDirty](#) ([AAX_CParamID](#) paramID, bool iDirty=true)
- [AAX_Result](#) [Dispatch](#) ()
- [AAX_Result](#) [GenerateSingleValuePacket](#) ([AAX_CParamID](#) iParam, [AAX_CPacket](#) &ioPacket)

14.26.2 Constructor & Destructor Documentation

14.26.2.1 AAX_CPacketDispatcher()

```
AAX_CPacketDispatcher::AAX_CPacketDispatcher ( )
```

14.26.2.2 ~AAX_CPacketDispatcher()

```
AAX_CPacketDispatcher::~~AAX_CPacketDispatcher ( )
```

14.26.3 Member Function Documentation

14.26.3.1 Initialize()

```
void AAX_CPacketDispatcher::Initialize (
    AAX\_IController * iPlugIn,
    AAX\_IEffectParameters * iEffectParameters )
```

14.26.3.2 RegisterPacket() [1/3]

```
AAX_Result AAX_CPacketDispatcher::RegisterPacket (
    AAX_CParamID paramID,
    AAX_CFieldIndex portID,
    const AAX_IPacketHandler * iHandler )
```

Referenced by [RegisterPacket\(\)](#).

Here is the caller graph for this function:



14.26.3.3 RegisterPacket() [2/3]

```
template<class TWorker , typename Func >
AAX_Result AAX_CPacketDispatcher::RegisterPacket (
    AAX_CParamID paramID,
    AAX_CFieldIndex portID,
    TWorker * iPt2Object,
    Func infPt ) [inline]
```

References [RegisterPacket\(\)](#).

Here is the call graph for this function:

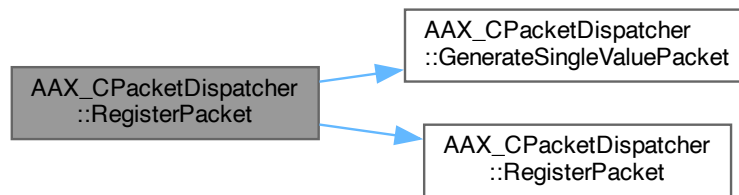


14.26.3.4 RegisterPacket() [3/3]

```
AAX_Result AAX_CPacketDispatcher::RegisterPacket (
    AAX_CParamID paramID,
    AAX_CFieldIndex portID ) [inline]
```

References [GenerateSingleValuePacket\(\)](#), and [RegisterPacket\(\)](#).

Here is the call graph for this function:

**14.26.3.5 SetDirty()**

```
AAX_Result AAX_CPacketDispatcher::SetDirty (
    AAX_CParamID paramID,
    bool iDirty = true )
```

14.26.3.6 Dispatch()

```
AAX_Result AAX_CPacketDispatcher::Dispatch ( )
```

14.26.3.7 GenerateSingleValuePacket()

```
AAX_Result AAX_CPacketDispatcher::GenerateSingleValuePacket (
    AAX_CParamID iParam,
    AAX_CPacket & ioPacket )
```

Referenced by [RegisterPacket\(\)](#).

Here is the caller graph for this function:



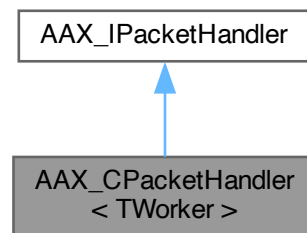
The documentation for this class was generated from the following file:

- [AAX_CPacketDispatcher.h](#)

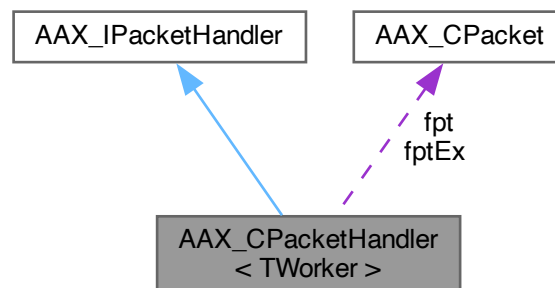
14.27 AAX_CPacketHandler< TWorker > Class Template Reference

```
#include <AAX_CPacketDispatcher.h>
```

Inheritance diagram for AAX_CPacketHandler< TWorker >:



Collaboration diagram for AAX_CPacketHandler< TWorker >:



14.27.1 Description

```
template<class TWorker>
class AAX_CPacketHandler< TWorker >
```

Callback container used by [AAX_CPacketDispatcher](#).

Public Member Functions

- [AAX_CPacketHandler](#) (TWorker *iPt2Object, fPt2Fn infPt)
- [AAX_CPacketHandler](#) (TWorker *iPt2Object, fPt2FnEx infPt)
- [AAX_IPacketHandler](#) * [Clone](#) () const
- [AAX_Result](#) Call ([AAX_CParamID](#) inParamID, [AAX_CPacket](#) &ioPacket) const

Public Member Functions inherited from [AAX_IPacketHandler](#)

- virtual [~AAX_IPacketHandler](#) ()
- virtual [AAX_IPacketHandler](#) * [Clone](#) () const =0
- virtual [AAX_Result](#) Call ([AAX_CParamID](#) inParamID, [AAX_CPacket](#) &ioPacket) const =0

Protected Attributes

- TWorker * [pt2Object](#)
- fPt2Fn [fpt](#)
- fPt2FnEx [fptEx](#)

14.27.2 Constructor & Destructor Documentation

14.27.2.1 AAX_CPacketHandler() [1/2]

```
template<class TWorker >
AAX_CPacketHandler< TWorker >::AAX_CPacketHandler (
    TWorker * iPt2Object,
    fPt2Fn infPt ) [inline]
```

14.27.2.2 AAX_CPacketHandler() [2/2]

```
template<class TWorker >
AAX_CPacketHandler< TWorker >::AAX_CPacketHandler (
    TWorker * iPt2Object,
    fPt2FnEx infPt ) [inline]
```

14.27.3 Member Function Documentation

14.27.3.1 Clone()

```
template<class TWorker >
AAX_IPacketHandler * AAX_CPacketHandler< TWorker >::Clone ( ) const [inline], [virtual]
```

Implements [AAX_IPacketHandler](#).

14.27.3.2 Call()

```
template<class TWorker >
AAX_Result AAX_CPacketHandler< TWorker >::Call (
    AAX_CParamID inParamID,
    AAX_CPacket & ioPacket ) const [inline], [virtual]
```

Implements [AAX_IPacketHandler](#).

References [AAX_ERROR_NULL_OBJECT](#), [AAX_CPacketHandler< TWorker >::fpt](#), [AAX_CPacketHandler< TWorker >::fptEx](#), and [AAX_CPacketHandler< TWorker >::pt2Object](#).

14.27.4 Member Data Documentation

14.27.4.1 pt2Object

```
template<class TWorker >
TWorker* AAX_CPacketHandler< TWorker >::pt2Object [protected]
```

Referenced by [AAX_CPacketHandler< TWorker >::Call\(\)](#).

14.27.4.2 fpt

```
template<class TWorker >
fPt2Fn AAX_CPacketHandler< TWorker >::fpt [protected]
```

Referenced by [AAX_CPacketHandler< TWorker >::Call\(\)](#).

14.27.4.3 fptEx

```
template<class TWorker >
fPt2FnEx AAX_CPacketHandler< TWorker >::fptEx [protected]
```

Referenced by [AAX_CPacketHandler< TWorker >::Call\(\)](#).

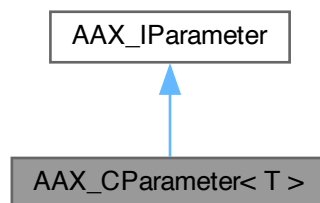
The documentation for this class was generated from the following file:

- [AAX_CPacketDispatcher.h](#)

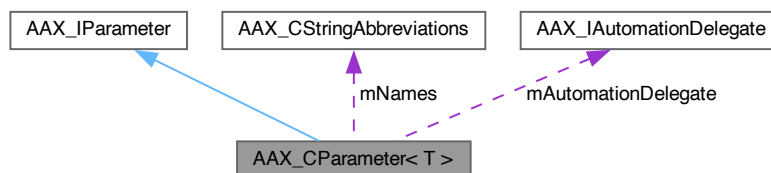
14.28 AAX_CParameter< T > Class Template Reference

```
#include <AAX_CParameter.h>
```

Inheritance diagram for AAX_CParameter< T >:



Collaboration diagram for AAX_CParameter< T >:



14.28.1 Description

```
template<typename T>
class AAX_CParameter< T >
```

Generic implementation of an [AAX_IParameter](#).

This is a concrete, templated implementation of [AAX_IParameter](#) for parameters with standard types such as `float`, `uint32`, `bool`, etc.

Many different behaviors can be composited into this class as delegates. [AAX_ITaperDelegate](#) and [AAX_IDisplayDelegate](#) are two examples of delegates that this class uses in order to apply custom behaviors to the [AAX_IParameter](#) interface.

Plug-in developers can subclass these delegates to create adaptable, reusable parameter behaviors, which can then be "mixed in" to individual [AAX_CParameter](#) objects without the need to modify the objects themselves.

Note

Because [AAX_CParameter](#) is a C++ template, each [AAX_CParameter](#) template parameter that is used creates a new subclass that adheres to the [AAX_IParameter](#) interface.

Public Types

- enum [Type](#) {
[eParameterTypeUndefined](#) = 0 ,
[eParameterTypeBool](#) = 1 ,
[eParameterTypeInt32](#) = 2 ,
[eParameterTypeFloat](#) = 3 ,
[eParameterTypeCustom](#) = 4 }
- enum [Defaults](#) {
[eParameterDefaultNumStepsDiscrete](#) = 2 ,
[eParameterDefaultNumStepsContinuous](#) = 128 }

Public Member Functions

- [AAX_CParameter](#) ([AAX_CParamID](#) identifier, const [AAX_IString](#) &name, T defaultValue, const [AAX_ITaperDelegate](#)< T > &taperDelegate, const [AAX_IDisplayDelegate](#)< T > &displayDelegate, bool automatable=false)
Constructs an [AAX_CParameter](#) object using the specified taper and display delegates.
- [AAX_CParameter](#) (const [AAX_IString](#) &identifier, const [AAX_IString](#) &name, T defaultValue, const [AAX_ITaperDelegate](#)< T > &taperDelegate, const [AAX_IDisplayDelegate](#)< T > &displayDelegate, bool automatable=false)
Constructs an [AAX_CParameter](#) object using the specified taper and display delegates.
- [AAX_CParameter](#) (const [AAX_IString](#) &identifier, const [AAX_IString](#) &name, T defaultValue, bool automatable=false)
Constructs an [AAX_CParameter](#) object with no delegates.
- [AAX_CParameter](#) (const [AAX_IString](#) &identifier, const [AAX_IString](#) &name, bool automatable=false)
Constructs an [AAX_CParameter](#) object with no delegates or default value.
- [AAX_DEFAULT_MOVE_CTOR](#) ([AAX_CParameter](#))
- [AAX_DEFAULT_MOVE_OPER](#) ([AAX_CParameter](#))
- [AAX_DELETE](#) ([AAX_CParameter](#)())
- [AAX_DELETE](#) ([AAX_CParameter](#)(const [AAX_CParameter](#) &other))
- [AAX_DELETE](#) ([AAX_CParameter](#) &operator=(const [AAX_CParameter](#) &other))

- [~AAX_CParameter \(\)](#) [AAX_OVERRIDE](#)
Virtual destructor used to delete all locally allocated pointers.
- [AAX_IParameterValue * CloneValue \(\)](#) const [AAX_OVERRIDE](#)
Clone the parameter's value to a new [AAX_IParameterValue](#) object.
- bool [GetValueAsString \(AAX_IString *\)](#) const
Retrieves the parameter's value as a string.
- bool [SetValueWithBool \(bool value\)](#)
Sets the parameter's value as a bool.
- bool [SetValueWithInt32 \(int32_t value\)](#)
Sets the parameter's value as an int32_t.
- bool [SetValueWithFloat \(float value\)](#)
Sets the parameter's value as a float.
- bool [SetValueWithDouble \(double value\)](#)
Sets the parameter's value as a double.
- bool [SetValueWithString \(const AAX_IString &value\)](#)
Sets the parameter's value as a string.
- bool [GetNormalizedValueFromBool \(bool value, double *normalizedValue\)](#) const
Converts a bool to a normalized parameter value.
- bool [GetNormalizedValueFromInt32 \(int32_t value, double *normalizedValue\)](#) const
Converts an integer to a normalized parameter value.
- bool [GetNormalizedValueFromFloat \(float value, double *normalizedValue\)](#) const
Converts a float to a normalized parameter value.
- bool [GetNormalizedValueFromDouble \(double value, double *normalizedValue\)](#) const
Converts a double to a normalized parameter value.
- bool [GetBoolFromNormalizedValue \(double inNormalizedValue, bool *value\)](#) const
Converts a normalized parameter value to a bool representing the corresponding real value.
- bool [GetInt32FromNormalizedValue \(double inNormalizedValue, int32_t *value\)](#) const
Converts a normalized parameter value to an integer representing the corresponding real value.
- bool [GetFloatFromNormalizedValue \(double inNormalizedValue, float *value\)](#) const
Converts a normalized parameter value to a float representing the corresponding real value.
- bool [GetDoubleFromNormalizedValue \(double inNormalizedValue, double *value\)](#) const
Converts a normalized parameter value to a double representing the corresponding real value.

Identification methods

- [AAX_CParamID Identifier \(\)](#) const [AAX_OVERRIDE](#)
Returns the parameter's unique identifier.
- void [SetName \(const AAX_CString &name\)](#) [AAX_OVERRIDE](#)
Sets the parameter's display name.
- const [AAX_CString & Name \(\)](#) const [AAX_OVERRIDE](#)
Returns the parameter's display name.
- void [AddShortenedName \(const AAX_CString &name\)](#) [AAX_OVERRIDE](#)
Sets the parameter's shortened display name.
- const [AAX_CString & ShortenedName \(int32_t iNumCharacters\)](#) const [AAX_OVERRIDE](#)
Returns the parameter's shortened display name.
- void [ClearShortenedNames \(\)](#) [AAX_OVERRIDE](#)
Clears the internal list of shortened display names.

Taper methods

- void [SetNormalizedDefaultValue \(double normalizedDefault\)](#) [AAX_OVERRIDE](#)
Sets the parameter's default value using its normalized representation.
- double [GetNormalizedDefaultValue \(\)](#) const [AAX_OVERRIDE](#)

- Returns the normalized representation of the parameter's real default value.*
- void [SetToDefaultValue](#) () [AAX_OVERRIDE](#)
Restores the state of this parameter to its default value.
- void [SetNormalizedValue](#) (double newNormalizedValue) [AAX_OVERRIDE](#)
Sets a parameter value using it's normalized representation.
- double [GetNormalizedValue](#) () const [AAX_OVERRIDE](#)
Returns the normalized representation of the parameter's current real value.
- void [SetNumberOfSteps](#) (uint32_t numSteps) [AAX_OVERRIDE](#)
Sets the number of discrete steps for this parameter.
- uint32_t [GetNumberOfSteps](#) () const [AAX_OVERRIDE](#)
Returns the number of discrete steps used by the parameter.
- uint32_t [GetStepValue](#) () const [AAX_OVERRIDE](#)
Returns the current step for the current value of the parameter.
- double [GetNormalizedValueFromStep](#) (uint32_t iStep) const [AAX_OVERRIDE](#)
Returns the normalized value for a given step.
- uint32_t [GetStepValueFromNormalizedValue](#) (double normalizedValue) const [AAX_OVERRIDE](#)
Returns the step value for a normalized value of the parameter.
- void [SetStepValue](#) (uint32_t iStep) [AAX_OVERRIDE](#)
Returns the current step for the current value of the parameter.
- void [SetType](#) ([AAX_EParameterType](#) iControlType) [AAX_OVERRIDE](#)
Sets the type of this parameter.
- [AAX_EParameterType](#) [GetType](#) () const [AAX_OVERRIDE](#)
Returns the type of this parameter as an [AAX_EParameterType](#).
- void [SetOrientation](#) ([AAX_EParameterOrientation](#) iOrientation) [AAX_OVERRIDE](#)
Sets the orientation of this parameter.
- [AAX_EParameterOrientation](#) [GetOrientation](#) () const [AAX_OVERRIDE](#)
Returns the orientation of this parameter.
- void [SetTaperDelegate](#) ([AAX_ITaperDelegateBase](#) &inTaperDelegate, bool inPreserveValue=true) [AAX_OVERRIDE](#)
Sets the parameter's taper delegate.

Display methods

- void [SetDisplayDelegate](#) ([AAX_IDisplayDelegateBase](#) &inDisplayDelegate) [AAX_OVERRIDE](#)
Sets the parameter's display delegate.
- bool [GetValueString](#) ([AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Serializes the parameter value into a string.
- bool [GetValueString](#) (int32_t iMaxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Serializes the parameter value into a string, size hint included.
- bool [GetNormalizedValueFromBool](#) (bool value, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a bool to a normalized parameter value.
- bool [GetNormalizedValueFromInt32](#) (int32_t value, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts an integer to a normalized parameter value.
- bool [GetNormalizedValueFromFloat](#) (float value, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a float to a normalized parameter value.
- bool [GetNormalizedValueFromDouble](#) (double value, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a double to a normalized parameter value.
- bool [GetNormalizedValueFromString](#) (const [AAX_CString](#) &valueString, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a given string to a normalized parameter value.
- bool [GetBoolFromNormalizedValue](#) (double normalizedValue, bool *value) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to a bool representing the corresponding real value.
- bool [GetInt32FromNormalizedValue](#) (double normalizedValue, int32_t *value) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to an integer representing the corresponding real value.
- bool [GetFloatFromNormalizedValue](#) (double normalizedValue, float *value) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to a float representing the corresponding real value.
- bool [GetDoubleFromNormalizedValue](#) (double normalizedValue, double *value) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to a double representing the corresponding real value.

- bool [GetStringFromNormalizedValue](#) (double normalizedValue, [AAX_CString](#) &valueString) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to a string representing the corresponding real value.
- bool [GetStringFromNormalizedValue](#) (double normalizedValue, int32_t iMaxNumChars, [AAX_CString](#) &valueString) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to a string representing the corresponding real, size hint included. value.
- bool [SetValueFromString](#) (const [AAX_CString](#) &newValueString) [AAX_OVERRIDE](#)
Converts a string to a real parameter value and sets the parameter to this value.

Automation methods

- void [SetAutomationDelegate](#) ([AAX_IAutomationDelegate](#) *iAutomationDelegate) [AAX_OVERRIDE](#)
Sets the automation delegate (if one is required)
- bool [Automatable](#) () const [AAX_OVERRIDE](#)
Returns true if the parameter is automatable, false if it is not.
- void [Touch](#) () [AAX_OVERRIDE](#)
Signals the automation system that a control has been touched.
- void [Release](#) () [AAX_OVERRIDE](#)
Signals the automation system that a control has been released.

Typed accessors

- bool [GetValueAsBool](#) (bool *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a bool.
- bool [GetValueAsInt32](#) (int32_t *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as an int32_t.
- bool [GetValueAsFloat](#) (float *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a float.
- bool [GetValueAsDouble](#) (double *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a double.
- bool [GetValueAsString](#) ([AAX_IString](#) *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a string.
- bool [SetValueWithBool](#) (bool value) [AAX_OVERRIDE](#)
Sets the parameter's value as a bool.
- bool [SetValueWithInt32](#) (int32_t value) [AAX_OVERRIDE](#)
Sets the parameter's value as an int32_t.
- bool [SetValueWithFloat](#) (float value) [AAX_OVERRIDE](#)
Sets the parameter's value as a float.
- bool [SetValueWithDouble](#) (double value) [AAX_OVERRIDE](#)
Sets the parameter's value as a double.
- bool [SetValueWithString](#) (const [AAX_IString](#) &value) [AAX_OVERRIDE](#)
Sets the parameter's value as a string.

Host interface methods

- void [UpdateNormalizedValue](#) (double newNormalizedValue) [AAX_OVERRIDE](#)
Sets the parameter's state given a normalized value.

Public Member Functions inherited from [AAX_IParameter](#)

- virtual [~AAX_IParameter](#) ()
Virtual destructor.
- virtual [AAX_IParameterValue](#) * [CloneValue](#) () const =0
Clone the parameter's value to a new [AAX_IParameterValue](#) object.

Direct methods on AAX_CParameter

These methods can be used to access the parameter's state and properties. These methods are specific to the concrete [AAX_CParameter](#) class and are not part of the [AAX_IParameter](#) interface.

- [AAX_CStringAbbreviations](#) mNames
- bool [mAutomatable](#)
- uint32_t [mNumSteps](#)
- [AAX_EParameterType](#) mControlType
- [AAX_EParameterOrientation](#) mOrientation
- [AAX_ITaperDelegate](#)< T > * [mTaperDelegate](#)
- [AAX_IDisplayDelegate](#)< T > * [mDisplayDelegate](#)
- [AAX_IAutomationDelegate](#) * [mAutomationDelegate](#)
- bool [mNeedNotify](#)
- [AAX_CParameterValue](#)< T > [mValue](#)
- T [mDefaultValue](#)
- void [SetValue](#) (T newValue)
Initiates a host request to set the parameter's value.
- T [GetValue](#) () const
Returns the parameter's value.
- void [SetDefaultValue](#) (T newDefaultValue)
Set the parameter's default value.
- T [GetDefaultValue](#) () const
Returns the parameter's default value.
- const [AAX_ITaperDelegate](#)< T > * [TaperDelegate](#) () const
Returns a reference to the parameter's taper delegate.
- const [AAX_IDisplayDelegate](#)< T > * [DisplayDelegate](#) () const
Returns a reference to the parameter's display delegate.

14.28.2 Member Enumeration Documentation

14.28.2.1 Type

```
template<typename T >
enum AAX\_CParameter::Type
```

Enumerator

eParameterTypeUndefined	
eParameterTypeBool	
eParameterTypeInt32	
eParameterTypeFloat	
eParameterTypeCustom	

14.28.2.2 Defaults

```
template<typename T >
enum AAX_CParameter::Defaults
```

Enumerator

eParameterDefaultNumStepsDiscrete	
eParameterDefaultNumStepsContinuous	

14.28.3 Constructor & Destructor Documentation

14.28.3.1 AAX_CParameter() [1/4]

```
template<typename T >
AAX_CParameter< T >::AAX_CParameter (
    AAX_CParamID identifier,
    const AAX_IString & name,
    T defaultValue,
    const AAX_ITaperDelegate< T > & taperDelegate,
    const AAX_IDisplayDelegate< T > & displayDelegate,
    bool automatable = false )
```

Constructs an [AAX_CParameter](#) object using the specified taper and display delegates.

The delegates are passed in by reference to prevent ambiguities of object ownership. For more information about `identifier` and `name`, please consult the base [AAX_IParameter](#) interface.

Parameters

in	<i>identifier</i>	Unique ID for the parameter, these can only be 31 characters long at most. (the fixed length is a requirement for some optimizations in the host)
in	<i>name</i>	The parameter's unabbreviated display name
in	<i>defaultValue</i>	The parameter's default value
in	<i>taperDelegate</i>	A delegate representing the parameter's taper behavior
in	<i>displayDelegate</i>	A delegate representing the parameter's display conversion behavior
in	<i>automatable</i>	A flag to set whether the parameter will be visible to the host's automation system

Note

Upon construction, the state (value) of the parameter will be the default value, as established by the provided `taperDelegate`.

Host Compatibility Notes As of Pro Tools 10.2, DAE will check for a matching parameter NAME and not an ID when reading in automation data from a session saved with an AAX plug-ins RTAS/↔ TDM counter part.

As of Pro Tools 11.1, AAE will first try to match ID. If that fails, AAE will fall back to matching by Name.

References [AAX_CParameter< T >::SetToDefaultValue\(\)](#).

Here is the call graph for this function:



14.28.3.2 AAX_CParameter() [2/4]

```

template<typename T >
AAX_CParameter< T >::AAX_CParameter (
    const AAX_IString & identifier,
    const AAX_IString & name,
    T defaultValue,
    const AAX_ITaperDelegate< T > & taperDelegate,
    const AAX_IDisplayDelegate< T > & displayDelegate,
    bool automatable = false )
  
```

Constructs an [AAX_CParameter](#) object using the specified taper and display delegates.

This constructor uses an [AAX_IString](#) for the parameter identifier, which can be a more flexible solution for some plug-ins.

References [AAX_CParameter< T >::SetToDefaultValue\(\)](#).

Here is the call graph for this function:



14.28.3.3 AAX_CParameter() [3/4]

```
template<typename T >
AAX_CParameter< T >::AAX_CParameter (
    const AAX_IString & identifier,
    const AAX_IString & name,
    T defaultValue,
    bool automatable = false )
```

Constructs an [AAX_CParameter](#) object with no delegates.

Delegates may be set on this object after construction. Most parameter operations will not work until after delegates have been set.

See also

- [AAX_CParameter::SetTaperDelegate\(\)](#)

See also

- [AAX_CParameter::SetDisplayDelegate\(\)](#)

References [AAX_CParameter< T >::SetToDefaultValue\(\)](#).

Here is the call graph for this function:

**14.28.3.4 AAX_CParameter()** [4/4]

```
template<typename T >
AAX_CParameter< T >::AAX_CParameter (
    const AAX_IString & identifier,
    const AAX_IString & name,
    bool automatable = false )
```

Constructs an [AAX_CParameter](#) object with no delegates or default value.

Delegates and default value may be set on this object after construction. Most parameter operations will not work until after delegates have been set.

See also

- [AAX_CParameter::SetDefaultValue\(\)](#)

See also

- [AAX_CParameter::SetTaperDelegate\(\)](#)

See also

- [AAX_CParameter::SetDisplayDelegate\(\)](#)

References [AAX_CParameter< T >::SetToDefaultValue\(\)](#).

Here is the call graph for this function:



14.28.3.5 ~AAX_CParameter()

```
template<typename T >
AAX_CParameter< T >::~~AAX_CParameter
```

Virtual destructor used to delete all locally allocated pointers.

14.28.4 Member Function Documentation

14.28.4.1 AAX_DEFAULT_MOVE_CTOR()

```
template<typename T >
AAX_CParameter< T >::AAX_DEFAULT_MOVE_CTOR (
    AAX_CParameter< T > )
```

Move constructor and move assignment operator are allowed

14.28.4.2 AAX_DEFAULT_MOVE_OPER()

```
template<typename T >
AAX_CParameter< T >::AAX_DEFAULT_MOVE_OPER (
    AAX_CParameter< T > )
```

14.28.4.3 AAX_DELETE() [1/3]

```
template<typename T >
AAX_CParameter< T >::AAX_DELETE (
    AAX_CParameter< T >() )
```

Default constructor not allowed, except by possible wrapper classes.

14.28.4.4 AAX_DELETE() [2/3]

```
template<typename T >
AAX_CParameter< T >::AAX_DELETE (
    AAX_CParameter< T >(const AAX_CParameter< T > &other) )
```

14.28.4.5 AAX_DELETE() [3/3]

```
template<typename T >
AAX_CParameter< T >::AAX_DELETE (
    AAX_CParameter< T > & operator = (const AAX_CParameter< T > &other) )
```

14.28.4.6 CloneValue()

```
template<typename T >
AAX_IParameterValue * AAX_CParameter< T >::CloneValue [virtual]
```

Clone the parameter's value to a new [AAX_IParameterValue](#) object.

The returned object is independent from the [AAX_IParameter](#). For example, changing the state of the returned object will not result in a change to the original [AAX_IParameter](#).

Implements [AAX_IParameter](#).

14.28.4.7 Identifier()

```
template<typename T >
AAX_CParamID AAX_CParameter< T >::Identifier [virtual]
```

Returns the parameter's unique identifier.

This unique ID is used by the [Parameter Manager](#) and by outside applications to uniquely identify and target control messages. This value may not be changed after the parameter has been constructed.

Implements [AAX_IParameter](#).

14.28.4.8 SetName()

```
template<typename T >
void AAX_CParameter< T >::SetName (
    const AAX_CString & name ) [virtual]
```

Sets the parameter's display name.

This name is used for display only, it is not used for indexing or identifying the parameter. This name may be changed after the parameter has been created, but display name changes may not be recognized by all AAX hosts.

Parameters

in	<i>name</i>	Display name that will be assigned to the parameter
----	-------------	---

Implements [AAX_IPParameter](#).

14.28.4.9 Name()

```
template<typename T >
const AAX\_CString & AAX\_CParameter< T >::Name [virtual]
```

Returns the parameter's display name.

Note

This method returns a const reference in order to prevent a string copy. Do not cast away the const to change this value.

Implements [AAX_IPParameter](#).

14.28.4.10 AddShortenedName()

```
template<typename T >
void AAX\_CParameter< T >::AddShortenedName (
    const AAX\_CString & name ) [virtual]
```

Sets the parameter's shortened display name.

This name is used for display only, it is not used for indexing or identifying the parameter. These names show up when the host asks for shorter length parameter names for display on Control Surfaces or other string length constrained situations.

Parameters

in	<i>name</i>	Shortened display names that will be assigned to the parameter
----	-------------	--

Implements [AAX_IPParameter](#).

14.28.4.11 ShortenedName()

```
template<typename T >
const AAX\_CString & AAX\_CParameter< T >::ShortenedName (
    int32_t iNumCharacters ) const [virtual]
```

Returns the parameter's shortened display name.

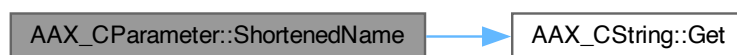
Note

This method returns a const reference in order to prevent a string copy. Do not cast away the const to change this value.

Implements [AAX_IParameter](#).

References [AAX_CString::Get\(\)](#).

Here is the call graph for this function:

**14.28.4.12 ClearShortenedNames()**

```
template<typename T >
void AAX_CParameter< T >::ClearShortenedNames [virtual]
```

Clears the internal list of shortened display names.

Implements [AAX_IParameter](#).

14.28.4.13 SetNormalizedDefaultValue()

```
template<typename T >
void AAX_CParameter< T >::SetNormalizedDefaultValue (
    double normalizedDefault ) [virtual]
```

Sets the parameter's default value using its normalized representation.

Implements [AAX_IParameter](#).

14.28.4.14 GetNormalizedDefaultValue()

```
template<typename T >
double AAX_CParameter< T >::GetNormalizedDefaultValue [virtual]
```

Returns the normalized representation of the parameter's real default value.

Implements [AAX_IParameter](#).

14.28.4.15 SetToDefaultValue()

```
template<typename T >
void AAX_CParameter< T >::SetToDefaultValue [virtual]
```

Restores the state of this parameter to its default value.

Implements [AAX_IPParameter](#).

Referenced by [AAX_CParameter< T >::AAX_CParameter\(\)](#).

Here is the caller graph for this function:

**14.28.4.16 SetNormalizedValue()**

```
template<typename T >
void AAX_CParameter< T >::SetNormalizedValue (
    double newNormalizedValue ) [virtual]
```

Sets a parameter value using it's normalized representation.

For more information regarding normalized values, see [Parameter Manager](#)

Parameters

in	<i>newNormalizedValue</i>	New value (normalized) to which the parameter will be set
----	---------------------------	---

Implements [AAX_IPParameter](#).

14.28.4.17 GetNormalizedValue()

```
template<typename T >
double AAX_CParameter< T >::GetNormalizedValue [virtual]
```

Returns the normalized representation of the parameter's current real value.

Implements [AAX_IPParameter](#).

14.28.4.18 SetNumberOfSteps()

```
template<typename T >
void AAX_CParameter< T >::SetNumberOfSteps (
    uint32_t numSteps ) [virtual]
```

Sets the number of discrete steps for this parameter.

Stepped parameter values are useful for discrete parameters and for "jumping" events such as mouse wheels, page up/down, etc. The parameter's step size is used to specify the coarseness of those changes.

Note

numSteps MUST be greater than zero. All other values may be considered an error by the host.

Parameters

in	<i>numSteps</i>	The number of steps that the parameter will use
----	-----------------	---

Implements [AAX_IParameter](#).

References [AAX_ASSERT](#).

14.28.4.19 GetNumberOfSteps()

```
template<typename T >
uint32_t AAX_CParameter< T >::GetNumberOfSteps [virtual]
```

Returns the number of discrete steps used by the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.28.4.20 GetStepValue()

```
template<typename T >
uint32_t AAX_CParameter< T >::GetStepValue [virtual]
```

Returns the current step for the current value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.28.4.21 GetNormalizedValueFromStep()

```
template<typename T >
double AAX_CParameter< T >::GetNormalizedValueFromStep (
    uint32_t iStep ) const [virtual]
```

Returns the normalized value for a given step.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.28.4.22 GetStepValueFromNormalizedValue()

```
template<typename T >
uint32_t AAX_CParameter< T >::GetStepValueFromNormalizedValue (
    double normalizedValue ) const [virtual]
```

Returns the step value for a normalized value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.28.4.23 SetStepValue()

```
template<typename T >
void AAX_CParameter< T >::SetStepValue (
    uint32_t iStep ) [virtual]
```

Returns the current step for the current value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.28.4.24 SetType()

```
template<typename T >
void AAX_CParameter< T >::SetType (
    AAX_EParameterType iControlType ) [virtual]
```

Sets the type of this parameter.

See [GetType](#) for use cases

Parameters

in	<i>iControlType</i>	The parameter's new type as an AAX_EParameterType
----	---------------------	---

Implements [AAX_IParameter](#).

14.28.4.25 GetType()

```
template<typename T >
AAX_EParameterType AAX_CParameter< T >::GetType [virtual]
```

Returns the type of this parameter as an AAX_EParameterType.

Todo Document use cases for control type

Implements [AAX_IParameter](#).

14.28.4.26 SetOrientation()

```
template<typename T >
void AAX_CParameter< T >::SetOrientation (
    AAX_EParameterOrientation iOrientation ) [virtual]
```

Sets the orientation of this parameter.

Parameters

in	<i>iOrientation</i>	The parameter's new orientation
----	---------------------	---------------------------------

Implements [AAX_IParameter](#).

14.28.4.27 GetOrientation()

```
template<typename T >
AAX_EParameterOrientation AAX_CParameter< T >::GetOrientation [virtual]
```

Returns the orientation of this parameter.

Implements [AAX_IParameter](#).

14.28.4.28 SetTaperDelegate()

```
template<typename T >
void AAX_CParameter< T >::SetTaperDelegate (
    AAX_ITaperDelegateBase & inTaperDelegate,
    bool inPreserveValue = true ) [virtual]
```

Sets the parameter's taper delegate.

Parameters

in	<i>inTaperDelegate</i>	A reference to the parameter's new taper delegate
in	<i>inPreserveValue</i>	

Todo Document this parameter

Implements [AAX_IParameter](#).

References [AAX_ITaperDelegate< T >::Clone\(\)](#).

Here is the call graph for this function:



14.28.4.29 SetDisplayDelegate()

```
template<typename T >
void AAX_CParameter< T >::SetDisplayDelegate (
    AAX_IDisplayDelegateBase & inDisplayDelegate ) [virtual]
```

Sets the parameter's display delegate.

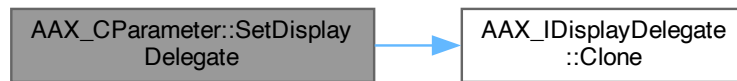
Parameters

in	<i>inDisplayDelegate</i>	A reference to the parameter's new display delegate
----	--------------------------	---

Implements [AAX_IParameter](#).

References [AAX_IDisplayDelegate< T >::Clone\(\)](#).

Here is the call graph for this function:



14.28.4.30 GetValueString() [1/2]

```

template<typename T >
bool AAX_CParameter< T >::GetValueString (
    AAX_CString * valueString ) const [virtual]
  
```

Serializes the parameter value into a string.

Parameters

out	<i>valueString</i>	A string representing the parameter's real value
-----	--------------------	--

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.31 GetValueString() [2/2]

```

template<typename T >
bool AAX_CParameter< T >::GetValueString (
    int32_t iMaxNumChars,
    AAX_CString * valueString ) const [virtual]
  
```

Serializes the parameter value into a string, size hint included.

Parameters

in	<i>iMaxNumChars</i>	A size hint for the size of the string being requested. Useful for control surfaces and other limited area text fields. (make sure that size of desired string also has room for null termination)
out	<i>valueString</i>	A string representing the parameter's real value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.32 GetNormalizedValueFromBool() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetNormalizedValueFromBool (
    bool value,
    double * normalizedValue ) const [virtual]
```

Converts a bool to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.33 GetNormalizedValueFromInt32() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetNormalizedValueFromInt32 (
    int32_t value,
    double * normalizedValue ) const [virtual]
```

Converts an integer to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
-------------	-------------------------------------

Return values

<i>false</i>	The value conversion was unsuccessful
--------------	---------------------------------------

Implements [AAX_IParameter](#).

14.28.4.34 GetNormalizedValueFromFloat() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetNormalizedValueFromFloat (
    float value,
    double * normalizedValue ) const [virtual]
```

Converts a float to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.35 GetNormalizedValueFromDouble() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetNormalizedValueFromDouble (
    double value,
    double * normalizedValue ) const [virtual]
```

Converts a double to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.36 GetNormalizedValueFromString()

```
template<typename T >
bool AAX_CParameter< T >::GetNormalizedValueFromString (
    const AAX_CString & valueString,
    double * normalizedValue ) const [virtual]
```

Converts a given string to a normalized parameter value.

Parameters

in	<i>valueString</i>	A string representing a possible real value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.37 GetBoolFromNormalizedValue() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetBoolFromNormalizedValue (
    double normalizedValue,
    bool * value ) const [virtual]
```

Converts a normalized parameter value to a bool representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.38 GetInt32FromNormalizedValue() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetInt32FromNormalizedValue (
    double normalizedValue,
    int32_t * value ) const [virtual]
```

Converts a normalized parameter value to an integer representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to int32_t was successful
<i>false</i>	The conversion to int32_t was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.39 GetFloatFromNormalizedValue() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetFloatFromNormalizedValue (
    double normalizedValue,
    float * value ) const [virtual]
```

Converts a normalized parameter value to a float representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.40 GetDoubleFromNormalizedValue() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetDoubleFromNormalizedValue (
```

```
double normalizedValue,
double * value ) const [virtual]
```

Converts a normalized parameter value to a double representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to double was successful
<i>false</i>	The conversion to double was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.41 GetStringFromNormalizedValue() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetStringFromNormalizedValue (
    double normalizedValue,
    AAX_CString & valueString ) const [virtual]
```

Converts a normalized parameter value to a string representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	A normalized parameter value
out	<i>valueString</i>	A string representing the parameter value associated with normalizedValue

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.42 GetStringFromNormalizedValue() [2/2]

```
template<typename T >
bool AAX_CParameter< T >::GetStringFromNormalizedValue (
    double normalizedValue,
    int32_t iMaxNumChars,
    AAX_CString & valueString ) const [virtual]
```

Converts a normalized parameter value to a string representing the corresponding real, size hint included. value.

Parameters

in	<i>normalizedValue</i>	A normalized parameter value
in	<i>iMaxNumChars</i>	A size hint for the size of the string being requested. Useful for control surfaces and other limited area text fields. (make sure that size of desired string also has room for null termination)
out	<i>valueString</i>	A string representing the parameter value associated with <i>normalizedValue</i>

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.43 SetValueFromString()

```
template<typename T >
bool AAX_CParameter< T >::SetValueFromString (
    const AAX_CString & newValueString ) [virtual]
```

Converts a string to a real parameter value and sets the parameter to this value.

Parameters

in	<i>newValueString</i>	A string representing the parameter's new real value
----	-----------------------	--

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.44 SetAutomationDelegate()

```
template<typename T >
void AAX_CParameter< T >::SetAutomationDelegate (
    AAX_IAutomationDelegate * iAutomationDelegate ) [virtual]
```

Sets the automation delegate (if one is required)

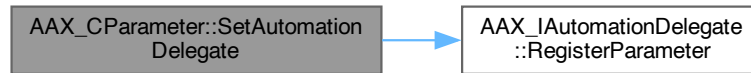
Parameters

in	<i>iAutomationDelegate</i>	A reference to the parameter manager's automation delegate interface
----	----------------------------	--

Implements [AAX_IParameter](#).

References [AAX_IAutomationDelegate::RegisterParameter\(\)](#).

Here is the call graph for this function:



14.28.4.45 Automatable()

```
template<typename T >
bool AAX_CParameter< T >::Automatable [virtual]
```

Returns true if the parameter is automatable, false if it is not.

Note

Subclasses that return true in this method must support host-based automation.

Implements [AAX_IParameter](#).

14.28.4.46 Touch()

```
template<typename T >
void AAX_CParameter< T >::Touch [virtual]
```

Signals the automation system that a control has been touched.

Call this method in response to GUI events that begin editing, such as a mouse down. After this method has been called you are free to call [SetNormalizedValue\(\)](#) as much as you need, e.g. in order to respond to subsequent mouse moved events. Call [Release\(\)](#) to free the parameter for updates from other controls.

Implements [AAX_IParameter](#).

14.28.4.47 Release()

```
template<typename T >
void AAX_CParameter< T >::Release [virtual]
```

Signals the automation system that a control has been released.

Call this method in response to GUI events that complete editing, such as a mouse up. Once this method has been called you should not call [SetNormalizedValue\(\)](#) again until after the next call to [Touch\(\)](#).

Implements [AAX_IParameter](#).

14.28.4.48 GetValueAsBool()

```
template<typename T >
bool AAX_CParameter< T >::GetValueAsBool (
    bool * value ) const [virtual]
```

Retrieves the parameter's value as a bool.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.49 GetValueAsInt32()

```
template<typename T >
bool AAX_CParameter< T >::GetValueAsInt32 (
    int32_t * value ) const [virtual]
```

Retrieves the parameter's value as an int32_t.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to int32_t was successful
-------------	--

Return values

<i>false</i>	The conversion to int32_t was unsuccessful
--------------	--

Implements [AAX_IParameter](#).

14.28.4.50 GetValueAsFloat()

```
template<typename T >
bool AAX_CParameter< T >::GetValueAsFloat (
    float * value ) const [virtual]
```

Retrieves the parameter's value as a float.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.51 GetValueAsDouble()

```
template<typename T >
bool AAX_CParameter< T >::GetValueAsDouble (
    double * value ) const [virtual]
```

Retrieves the parameter's value as a double.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to double was successful
<i>false</i>	The conversion to double was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.52 GetValueAsString() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::GetValueAsString (
    AAX_IString * value ) const [virtual]
```

Retrieves the parameter's value as a string.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to string was successful
<i>false</i>	The conversion to string was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.53 SetValueWithBool() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::SetValueWithBool (
    bool value ) [virtual]
```

Sets the parameter's value as a bool.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from bool was successful
<i>false</i>	The conversion from bool was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.54 SetValueWithInt32() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::SetValueWithInt32 (
    int32_t value ) [virtual]
```

Sets the parameter's value as an int32_t.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion from int32_t was successful
false	The conversion from int32_t was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.55 SetValueWithFloat() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::SetValueWithFloat (
    float value ) [virtual]
```

Sets the parameter's value as a float.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion from float was successful
false	The conversion from float was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.56 SetValueWithDouble() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::SetValueWithDouble (
    double value ) [virtual]
```

Sets the parameter's value as a double.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

<i>true</i>	The conversion from double was successful
<i>false</i>	The conversion from double was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.57 SetValueWithString() [1/2]

```
template<typename T >
bool AAX_CParameter< T >::SetValueWithString (
    const AAX_IString & value ) [virtual]
```

Sets the parameter's value as a string.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from string was successful
<i>false</i>	The conversion from string was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.58 UpdateNormalizedValue()

```
template<typename T >
void AAX_CParameter< T >::UpdateNormalizedValue (
    double newNormalizedValue ) [virtual]
```

Sets the parameter's state given a normalized value.

This is the second half of the parameter setting operation that is initiated with a call to [SetValue\(\)](#). Parameters should not be set directly using this method; instead, use [SetValue\(\)](#).

Parameters

in	<i>newNormalizedValue</i>	Normalized value that will be used to set the parameter's new state
----	---------------------------	---

Implements [AAX_IParameter](#).

14.28.4.59 SetValue()

```
template<typename T >
void AAX_CParameter< T >::SetValue (
    T newValue )
```

Initiates a host request to set the parameter's value.

This method normalizes the provided value and sends a request for the value change to the AAX host. The host responds with a call to [AAX_IPParameter::UpdateNormalizedValue\(\)](#) to complete the set operation.

Parameters

in	<i>newValue</i>	The parameter's new value
----	-----------------	---------------------------

14.28.4.60 GetValue()

```
template<typename T >
T AAX_CParameter< T >::GetValue
```

Returns the parameter's value.

This is the parameter's real, logical value and should not be normalized

14.28.4.61 SetDefaultValue()

```
template<typename T >
void AAX_CParameter< T >::SetDefaultValue (
    T newDefaultValue )
```

Set the parameter's default value.

This is the parameter's real, logical value and should not be normalized

Parameters

in	<i>newDefaultValue</i>	The parameter's new default value
----	------------------------	-----------------------------------

14.28.4.62 GetDefaultValue()

```
template<typename T >
T AAX_CParameter< T >::GetDefaultValue
```

Returns the parameter's default value.

This is the parameter's real, logical value and should not be normalized

14.28.4.63 TaperDelegate()

```
template<typename T >
const AAX_ITaperDelegate< T > * AAX_CParameter< T >::TaperDelegate
```

Returns a reference to the parameter's taper delegate.

14.28.4.64 DisplayDelegate()

```
template<typename T >
const AAX_IDisplayDelegate< T > * AAX_CParameter< T >::DisplayDelegate
```

Returns a reference to the parameter's display delegate.

14.28.4.65 GetValueAsString() [2/2]

```
bool AAX_CParameter< AAX_CString >::GetValueAsString (
    AAX_IString * value ) const [virtual]
```

Retrieves the parameter's value as a string.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to string was successful
<i>false</i>	The conversion to string was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.66 SetValueWithBool() [2/2]

```
bool AAX_CParameter< bool >::SetValueWithBool (
    bool value ) [virtual]
```

Sets the parameter's value as a bool.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from bool was successful
<i>false</i>	The conversion from bool was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.67 SetValueWithInt32() [2/2]

```
bool AAX_CParameter< int32_t >::SetValueWithInt32 (
    int32_t value ) [virtual]
```

Sets the parameter's value as an int32_t.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from int32_t was successful
<i>false</i>	The conversion from int32_t was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.68 SetValueWithFloat() [2/2]

```
bool AAX_CParameter< float >::SetValueWithFloat (
    float value ) [virtual]
```

Sets the parameter's value as a float.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from float was successful
<i>false</i>	The conversion from float was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.69 SetValueWithDouble() [2/2]

```
bool AAX_CParameter< double >::SetValueWithDouble (
    double value ) [virtual]
```

Sets the parameter's value as a double.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from double was successful
<i>false</i>	The conversion from double was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.70 SetValueWithString() [2/2]

```
bool AAX_CParameter< AAX_CString >::SetValueWithString (
    const AAX_IString & value ) [virtual]
```

Sets the parameter's value as a string.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from string was successful
<i>false</i>	The conversion from string was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.71 GetNormalizedValueFromBool() [2/2]

```
bool AAX_CParameter< bool >::GetNormalizedValueFromBool (
    bool value,
    double * normalizedValue ) const [virtual]
```

Converts a bool to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.72 GetNormalizedValueFromInt32() [2/2]

```
bool AAX_CParameter< int32_t >::GetNormalizedValueFromInt32 (
    int32_t value,
    double * normalizedValue ) const [virtual]
```

Converts an integer to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.73 GetNormalizedValueFromFloat() [2/2]

```
bool AAX_CParameter< float >::GetNormalizedValueFromFloat (
    float value,
    double * normalizedValue ) const [virtual]
```

Converts a float to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.74 GetNormalizedValueFromDouble() [2/2]

```
bool AAX_CParameter< double >::GetNormalizedValueFromDouble (
    double value,
    double * normalizedValue ) const [virtual]
```

Converts a double to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.75 GetBoolFromNormalizedValue() [2/2]

```
bool AAX_CParameter< bool >::GetBoolFromNormalizedValue (
    double normalizedValue,
    bool * value ) const [virtual]
```

Converts a normalized parameter value to a bool representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.76 GetInt32FromNormalizedValue() [2/2]

```
bool AAX_CParameter< int32_t >::GetInt32FromNormalizedValue (
    double normalizedValue,
    int32_t * value ) const [virtual]
```

Converts a normalized parameter value to an integer representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to int32_t was successful
<i>false</i>	The conversion to int32_t was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.77 GetFloatFromNormalizedValue() [2/2]

```
bool AAX_CParameter< float >::GetFloatFromNormalizedValue (
    double normalizedValue,
    float * value ) const [virtual]
```

Converts a normalized parameter value to a float representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implements [AAX_IParameter](#).

14.28.4.78 GetDoubleFromNormalizedValue() [2/2]

```
bool AAX_CParameter< double >::GetDoubleFromNormalizedValue (
    double normalizedValue,
    double * value ) const [virtual]
```

Converts a normalized parameter value to a double representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to double was successful
<i>false</i>	The conversion to double was unsuccessful

Implements [AAX_IParameter](#).

14.28.5 Member Data Documentation

14.28.5.1 mName

```
template<typename T >
AAX_CStringAbbreviations AAX_CParameter< T >::mName [protected]
```

14.28.5.2 mAutomatable

```
template<typename T >
bool AAX_CParameter< T >::mAutomatable [protected]
```

14.28.5.3 mNumSteps

```
template<typename T >
uint32_t AAX_CParameter< T >::mNumSteps [protected]
```

14.28.5.4 mControlType

```
template<typename T >  
AAX_EParameterType AAX_CParameter< T >::mControlType [protected]
```

14.28.5.5 mOrientation

```
template<typename T >  
AAX_EParameterOrientation AAX_CParameter< T >::mOrientation [protected]
```

14.28.5.6 mTaperDelegate

```
template<typename T >  
AAX_ITaperDelegate<T>* AAX_CParameter< T >::mTaperDelegate [protected]
```

14.28.5.7 mDisplayDelegate

```
template<typename T >  
AAX_IDisplayDelegate<T>* AAX_CParameter< T >::mDisplayDelegate [protected]
```

14.28.5.8 mAutomationDelegate

```
template<typename T >  
AAX_IAutomationDelegate* AAX_CParameter< T >::mAutomationDelegate [protected]
```

14.28.5.9 mNeedNotify

```
template<typename T >  
bool AAX_CParameter< T >::mNeedNotify [protected]
```

14.28.5.10 mValue

```
template<typename T >  
AAX_CParameterValue<T> AAX_CParameter< T >::mValue [protected]
```


14.28.5.11 mDefaultValue

```
template<typename T >
T AAX_CParameter< T >::mDefaultValue [protected]
```

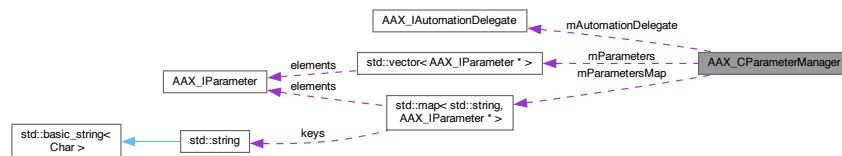
The documentation for this class was generated from the following file:

- [AAX_CParameter.h](#)

14.29 AAX_CParameterManager Class Reference

```
#include <AAX_CParameterManager.h>
```

Collaboration diagram for AAX_CParameterManager:



14.29.1 Description

A container object for plug-in parameters.

This implementation uses a STL vector to store a plug-in's set of parameters. This class contains a real implementation of the [Parameter Manager](#) (as opposed to a proxy.)

For more information, see [Parameter Manager](#).

Todo Should the Parameter Manager return error codes?

Public Member Functions

- [AAX_CParameterManager](#) ()
- [~AAX_CParameterManager](#) ()
- void [Initialize](#) ([AAX_IAutomationDelegate](#) *iAutomationDelegateUnknown)
Initialize the parameter manager.
- int32_t [NumParameters](#) () const
Returns the number of parameters in this instance of the parameter manager.
- void [RemoveParameterByID](#) ([AAX_CParamID](#) identifier)
Removes a parameter from the manager.
- void [RemoveAllParameters](#) ()
Removes all parameters from the manager.
- [AAX_IParameter](#) * [GetParameterByID](#) ([AAX_CParamID](#) identifier)

- Given a parameter ID, retrieves a reference to the requested parameter.*
- `const AAX_IPParameter * GetParameterByID (AAX_CParamID identifier) const`
- Given a parameter ID, retrieves a const reference to the requested parameter.*
- `AAX_IPParameter * GetParameterByName (const char *name)`
- Given a parameter name, retrieves a reference to the requested parameter.*
- `const AAX_IPParameter * GetParameterByName (const char *name) const`
- Given a parameter name, retrieves a const reference to the requested parameter.*
- `AAX_IPParameter * GetParameter (int32_t index)`
- Given a parameter index, retrieves a reference to the requested parameter.*
- `const AAX_IPParameter * GetParameter (int32_t index) const`
- Given a parameter index, retrieves a const reference to the requested parameter.*
- `int32_t GetParameterIndex (AAX_CParamID identifier) const`
- `void AddParameter (AAX_IPParameter *param)`
- `void RemoveParameter (AAX_IPParameter *param)`

Protected Attributes

- `AAX_IAutomationDelegate * mAutomationDelegate`
- `std::vector< AAX_IPParameter * > mParameters`
- `std::map< std::string, AAX_IPParameter * > mParametersMap`

14.29.2 Constructor & Destructor Documentation

14.29.2.1 AAX_CParameterManager()

```
AAX_CParameterManager::AAX_CParameterManager ( )
```

14.29.2.2 ~AAX_CParameterManager()

```
AAX_CParameterManager::~~AAX_CParameterManager ( )
```

14.29.3 Member Function Documentation

14.29.3.1 Initialize()

```
void AAX_CParameterManager::Initialize (
    AAX_IAutomationDelegate * iAutomationDelegateUnknown )
```

Initialize the parameter manager.

Called when plug-in instance is first instantiated. This method will initialize the plug-in's automation delegate, among other set-up tasks.

Parameters

in	<i>iAutomationDelegateUnknown</i>	A reference to the plug-in's AAX_IAutomationDelegate interface
----	-----------------------------------	--

14.29.3.2 NumParameters()

```
int32_t AAX_CParameterManager::NumParameters ( ) const
```

Returns the number of parameters in this instance of the parameter manager.

14.29.3.3 RemoveParameterByID()

```
void AAX_CParameterManager::RemoveParameterByID (
    AAX_CParamID identifier )
```

Removes a parameter from the manager.

Todo Should this method return success/failure code?

Parameters

in	<i>identifier</i>	ID of the parameter that will be removed
----	-------------------	--

14.29.3.4 RemoveAllParameters()

```
void AAX_CParameterManager::RemoveAllParameters ( )
```

Removes all parameters from the manager.

Todo Should this method return success/failure code?

14.29.3.5 GetParameterByID() [1/2]

```
AAX_IParameter * AAX_CParameterManager::GetParameterByID (
    AAX_CParamID identifier )
```

Given a parameter ID, retrieves a reference to the requested parameter.

Parameters

in	<i>identifier</i>	ID of the parameter that will be retrieved
----	-------------------	--

Referenced by [AAX_CMonolithicParameters::UpdateParameterNormalizedValue\(\)](#).

Here is the caller graph for this function:



14.29.3.6 GetParameterByID() [2/2]

```
const AAX_IParameter * AAX_CParameterManager::GetParameterByID (
    AAX_CParamID identifier ) const
```

Given a parameter ID, retrieves a const reference to the requested parameter.

Parameters

in	<i>identifier</i>	ID of the parameter that will be retrieved
----	-------------------	--

14.29.3.7 GetParameterByName() [1/2]

```
AAX_IParameter * AAX_CParameterManager::GetParameterByName (
    const char * name )
```

Given a parameter name, retrieves a reference to the requested parameter.

Note

Parameter names may be ambiguous

Parameters

in	<i>name</i>	Name of the parameter that will be retrieved
----	-------------	--

14.29.3.8 GetParameterByName() [2/2]

```
const AAX_IParameter * AAX_CParameterManager::GetParameterByName (
    const char * name ) const
```

Given a parameter name, retrieves a const reference to the requested parameter.

Note

Parameter names may be ambiguous

Parameters

in	<i>name</i>	ID of the parameter that will be retrieved
----	-------------	--

14.29.3.9 GetParameter() [1/2]

```
AAX_IParameter * AAX_CParameterManager::GetParameter (
    int32_t index )
```

Given a parameter index, retrieves a reference to the requested parameter.

Parameter indices are incremented in the order that parameters are added to the manager. See [AddParameter\(\)](#).

Parameters

in	<i>index</i>	Index of the parameter that will be retrieved
----	--------------	---

14.29.3.10 GetParameter() [2/2]

```
const AAX_IParameter * AAX_CParameterManager::GetParameter (
    int32_t index ) const
```

Given a parameter index, retrieves a const reference to the requested parameter.

Parameter indices are incremented in the order that parameters are added to the manager. See [AddParameter\(\)](#).

Parameters

in	<i>index</i>	Index of the parameter that will be retrieved
----	--------------	---

14.29.3.11 GetParameterIndex()

```
int32_t AAX_CParameterManager::GetParameterIndex (
    AAX_CParamID identifier ) const
```

Given a parameter ID, retrieves the index for the specified parameter

Parameters

in	<i>identifier</i>	ID of the parameter that will be retrieved
----	-------------------	--

14.29.3.12 AddParameter()

```
void AAX_CParameterManager::AddParameter (
    AAX_IParameter * param )
```

Adds a parameter to the manager

Todo Should this method return success/failure code?

Parameters

in	<i>param</i>	Reference to the parameter that will be added
----	--------------	---

14.29.3.13 RemoveParameter()

```
void AAX_CParameterManager::RemoveParameter (
    AAX_IParameter * param )
```

Removes a parameter to the manager

Todo Should this method return success/failure code?

Parameters

in	<i>param</i>	Reference to the parameter that will be removed
----	--------------	---

14.29.4 Member Data Documentation

14.29.4.1 mAutomationDelegate

[AAX_IAutomationDelegate*](#) AAX_CParameterManager::mAutomationDelegate [protected]

14.29.4.2 mParameters

std::vector<[AAX_IParameter*](#)> AAX_CParameterManager::mParameters [protected]

14.29.4.3 mParametersMap

std::map<std::string, [AAX_IParameter*](#)> AAX_CParameterManager::mParametersMap [protected]

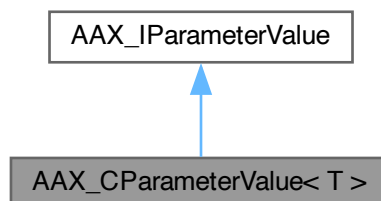
The documentation for this class was generated from the following file:

- [AAX_CParameterManager.h](#)

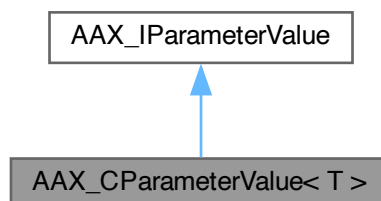
14.30 AAX_CParameterValue< T > Class Template Reference

```
#include <AAX_CParameter.h>
```

Inheritance diagram for AAX_CParameterValue< T >:



Collaboration diagram for AAX_CParameterValue< T >:



14.30.1 Description

```
template<typename T>
class AAX_CParameterValue< T >
```

Concrete implementation of [AAX_IParameterValue](#).

Used by [AAX_CParameter](#)

Public Types

- enum [Defaults](#) {
[eParameterDefaultMaxIdentifierSize](#) = kAAX_ParameterIdentifierMaxSize ,
[eParameterDefaultMaxIdentifierLength](#) = eParameterDefaultMaxIdentifierSize - 1 }

Public Member Functions

- [AAX_DEFAULT_DTOR_OVERRIDE](#) ([AAX_CParameterValue](#))
- [AAX_DEFAULT_MOVE_CTOR](#) ([AAX_CParameterValue](#))
- [AAX_DEFAULT_MOVE_OPER](#) ([AAX_CParameterValue](#))
- [AAX_DELETE](#) ([AAX_CParameterValue](#) &operator=(const [AAX_CParameterValue](#) &))
- [AAX_CParameterValue](#) ([AAX_CParamID](#) identifier)
Constructs an [AAX_CParameterValue](#) object.
- [AAX_CParameterValue](#) ([AAX_CParamID](#) identifier, const T &value)
Constructs an [AAX_CParameterValue](#) object with a defined initial state.
- [AAX_CParameterValue](#) (const [AAX_CParameterValue](#)< T > &other)
Copy constructor for [AAX_CParameterValue](#).
- const T & [Get](#) () const
Direct access to the template instance's value.
- void [Set](#) (const T &inValue)
Direct access to the template instance's value.
- [AAX_IParameterValue](#) * [Clone](#) () const [AAX_OVERRIDE](#)
Clones the parameter object.
- [AAX_CParamID](#) [Identifier](#) () const [AAX_OVERRIDE](#)
Returns the parameter's unique identifier.
- bool [GetValueAsBool](#) (bool *value) const
Retrieves the parameter's value as a bool.
- bool [GetValueAsInt32](#) (int32_t *value) const
Retrieves the parameter's value as an int32_t.
- bool [GetValueAsFloat](#) (float *value) const
Retrieves the parameter's value as a float.
- bool [GetValueAsDouble](#) (double *value) const
Retrieves the parameter's value as a double.
- bool [GetValueAsString](#) ([AAX_IString](#) *value) const
Retrieves the parameter's value as a string.

Public Member Functions inherited from [AAX_IParаметerValue](#)

- virtual [~AAX_IParаметerValue](#) ()
Virtual destructor.
- virtual [AAX_IParаметerValue * Clone](#) () const =0
Clones the parameter object.
- virtual [AAX_CParamID Identifier](#) () const =0
Returns the parameter's unique identifier.

Typed accessors

- bool [GetValueAsBool](#) (bool *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a bool.
- bool [GetValueAsInt32](#) (int32_t *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as an int32_t.
- bool [GetValueAsFloat](#) (float *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a float.
- bool [GetValueAsDouble](#) (double *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a double.
- bool [GetValueAsString](#) ([AAX_IString](#) *value) const [AAX_OVERRIDE](#)
Retrieves the parameter's value as a string.

14.30.2 Member Enumeration Documentation**14.30.2.1 Defaults**

```
template<typename T >
enum AAX\_CParameterValue::Defaults
```

Enumerator

eParameterDefaultMaxIdentifierSize	
eParameterDefaultMaxIdentifierLength	

14.30.3 Constructor & Destructor Documentation**14.30.3.1 [AAX_CParameterValue](#)() [1/3]**

```
template<typename T >
AAX\_CParameterValue< T >::AAX\_CParameterValue (
    AAX\_CParamID identifier ) [explicit]
```

Constructs an [AAX_CParameterValue](#) object.

Parameters

in	<i>identifier</i>	Unique ID for the parameter value, these can only be 31 characters long at most. (the fixed length is a requirement for some optimizations in the host)
----	-------------------	---

Note

The initial state of the parameter value is undefined

14.30.3.2 AAX_CParameterValue() [2/3]

```
template<typename T >
AAX_CParameterValue< T >::AAX_CParameterValue (
    AAX_CParamID identifier,
    const T & value ) [explicit]
```

Constructs an [AAX_CParameterValue](#) object with a defined initial state.

Parameters

in	<i>identifier</i>	Unique ID for the parameter value, these can only be 31 characters long at most. (the fixed length is a requirement for some optimizations in the host)
in	<i>value</i>	Initial state of the parameter value

14.30.3.3 AAX_CParameterValue() [3/3]

```
template<typename T >
AAX_CParameterValue< T >::AAX_CParameterValue (
    const AAX_CParameterValue< T > & other ) [explicit]
```

Copy constructor for [AAX_CParameterValue](#).

14.30.4 Member Function Documentation**14.30.4.1 AAX_DEFAULT_DTOR_OVERRIDE()**

```
template<typename T >
AAX_CParameterValue< T >::AAX_DEFAULT_DTOR_OVERRIDE (
    AAX_CParameterValue< T > )
```

14.30.4.2 AAX_DEFAULT_MOVE_CTOR()

```
template<typename T >
AAX_CParameterValue< T >::AAX_DEFAULT_MOVE_CTOR (
    AAX_CParameterValue< T > )
```

14.30.4.3 AAX_DEFAULT_MOVE_OPER()

```
template<typename T >
AAX_CParameterValue< T >::AAX_DEFAULT_MOVE_OPER (
    AAX_CParameterValue< T > )
```

14.30.4.4 AAX_DELETE()

```
template<typename T >
AAX_CParameterValue< T >::AAX_DELETE (
    AAX_CParameterValue< T > & operator = (const AAX_CParameterValue< T > &) )
```

14.30.4.5 Get()

```
template<typename T >
const T & AAX_CParameterValue< T >::Get ( ) const [inline]
```

Direct access to the template instance's value.

14.30.4.6 Set()

```
template<typename T >
void AAX_CParameterValue< T >::Set (
    const T & inValue ) [inline]
```

Direct access to the template instance's value.

14.30.4.7 Clone()

```
template<typename T >
AAX_IParparameterValue * AAX_CParameterValue< T >::Clone ( ) const [inline], [virtual]
```

Clones the parameter object.

Note

Does NOT set the automation delegate on the clone; ownership of the automation delegate and parameter registration/unregistration stays with the original parameter

Implements [AAX_IParparameterValue](#).

14.30.4.8 Identifier()

```
template<typename T >
AAX_CParamID AAX_CParameterValue< T >::Identifier ( ) const [inline], [virtual]
```

Returns the parameter's unique identifier.

This unique ID is used by the [Parameter Manager](#) and by outside applications to uniquely identify and target control messages. This value may not be changed after the parameter has been constructed.

Implements [AAX_IParparameterValue](#).

14.30.4.9 GetValueAsBool() [1/2]

```
template<typename T >
bool AAX_CParameterValue< T >::GetValueAsBool (
    bool * value ) const [virtual]
```

Retrieves the parameter's value as a bool.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implements [AAX_IParparameterValue](#).

14.30.4.10 GetValueAsInt32() [1/2]

```
template<typename T >
bool AAX_CParameterValue< T >::GetValueAsInt32 (
    int32_t * value ) const [virtual]
```

Retrieves the parameter's value as an int32_t.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to int32_t was successful
false	The conversion to int32_t was unsuccessful

Implements [AAX_IPParameterValue](#).

14.30.4.11 GetValueAsFloat() [1/2]

```
template<typename T >
bool AAX_CParameterValue< T >::GetValueAsFloat (
    float * value ) const [virtual]
```

Retrieves the parameter's value as a float.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to float was successful
false	The conversion to float was unsuccessful

Implements [AAX_IPParameterValue](#).

14.30.4.12 GetValueAsDouble() [1/2]

```
template<typename T >
bool AAX_CParameterValue< T >::GetValueAsDouble (
    double * value ) const [virtual]
```

Retrieves the parameter's value as a double.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to double was successful
false	The conversion to double was unsuccessful

Implements [AAX_IParаметerValue](#).

14.30.4.13 GetValueAsString() [1/2]

```
template<typename T >
bool AAX_CParameterValue< T >::GetValueAsString (
    AAX_IString * value ) const [virtual]
```

Retrieves the parameter's value as a string.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to string was successful
false	The conversion to string was unsuccessful

Implements [AAX_IParаметerValue](#).

14.30.4.14 GetValueAsBool() [2/2]

```
bool AAX_CParameterValue< bool >::GetValueAsBool (
    bool * value ) const [virtual]
```

Retrieves the parameter's value as a bool.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to bool was successful
------	---------------------------------------

Return values

<i>false</i>	The conversion to bool was unsuccessful
--------------	---

Implements [AAX_IParameterValue](#).

14.30.4.15 GetValueAsInt32() [2/2]

```
bool AAX_CParameterValue< int32_t >::GetValueAsInt32 (
    int32_t * value ) const [virtual]
```

Retrieves the parameter's value as an `int32_t`.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to <code>int32_t</code> was successful
<i>false</i>	The conversion to <code>int32_t</code> was unsuccessful

Implements [AAX_IParameterValue](#).

14.30.4.16 GetValueAsFloat() [2/2]

```
bool AAX_CParameterValue< float >::GetValueAsFloat (
    float * value ) const [virtual]
```

Retrieves the parameter's value as a float.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implements [AAX_IParameterValue](#).

14.30.4.17 GetValueAsDouble() [2/2]

```
bool AAX_CParameterValue< double >::GetValueAsDouble (
    double * value ) const [virtual]
```

Retrieves the parameter's value as a double.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to double was successful
false	The conversion to double was unsuccessful

Implements [AAX_IParаметerValue](#).

14.30.4.18 GetValueAsString() [2/2]

```
bool AAX_CParameterValue< AAX_CString >::GetValueAsString (
    AAX_IString * value ) const [virtual]
```

Retrieves the parameter's value as a string.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to string was successful
false	The conversion to string was unsuccessful

Implements [AAX_IParаметerValue](#).

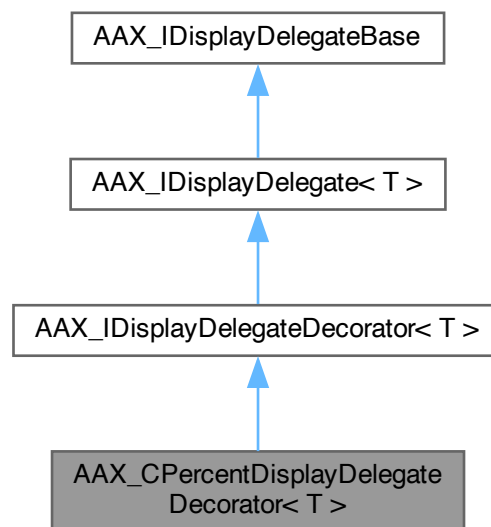
The documentation for this class was generated from the following file:

- [AAX_CParameter.h](#)

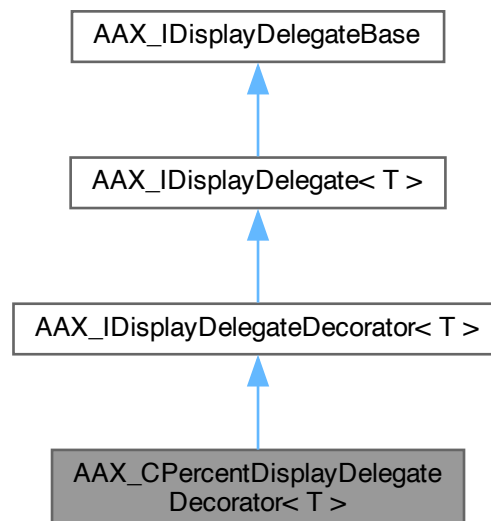
14.31 AAX_CPercentDisplayDelegateDecorator< T > Class Template Reference

```
#include <AAX_CPercentDisplayDelegateDecorator.h>
```


Inheritance diagram for AAX_CPercentDisplayDelegateDecorator< T >:



Collaboration diagram for AAX_CPercentDisplayDelegateDecorator< T >:



14.31.1 Description

```
template<typename T>
class AAX_CPercentDisplayDelegateDecorator< T >
```

A percent decorator conforming to [AAX_IDisplayDelegateDecorator](#).

This class is an [AAX_IDisplayDelegateDecorator](#), meaning that it acts as a wrapper for other display delegates or concrete display types. For more information about display delegate decorators in [AAX](#), see [Display delegate decorators](#)

The behavior of this class is to provide string conversion to and from percentage (%) values. When converting a parameter value to a string, it takes the real value and performs a % conversion before passing the value on to a concrete implementation to get a value string. It then adds on the "%" string at the end to signify that the value was converted. This allows something like a gain value to remain internally linear at all times even though its display is converted to a percentage.

The inverse operation is also supported; this class can convert a percentage-formatted string into its associated real value. The string will first be converted to a number, then that number will have the inverse % calculation applied to it to retrieve the parameter's actual value.

Public Member Functions

- [AAX_CPercentDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate)
- [AAX_CPercentDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateDecorator](#)< T >

- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate)
Constructor.
- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegateDecorator](#) &other)
Copy constructor.
- [~AAX_IDisplayDelegateDecorator](#) () [AAX_OVERRIDE](#)
Virtual destructor.
- [AAX_IDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate decorator.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value with a size constraint.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.31.2 Constructor & Destructor Documentation**14.31.2.1 AAX_CPercentDisplayDelegateDecorator()**

```
template<typename T >
AAX_CPercentDisplayDelegateDecorator< T >::AAX_CPercentDisplayDelegateDecorator (
    const AAX\_IDisplayDelegate< T > & displayDelegate )
```

14.31.3 Member Function Documentation**14.31.3.1 Clone()**

```
template<typename T >
AAX_CPercentDisplayDelegateDecorator< T > * AAX\_CPercentDisplayDelegateDecorator< T >::Clone
( ) const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>*    AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.31.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX\_CPercentDisplayDelegateDecorator< T >::ValueToString (
    T value,
    AAX\_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to <i>value</i>

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:



14.31.3.3 ValueToString() [2/2]

```

template<typename T >
bool AAX_CPercentDisplayDelegateDecorator< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
  
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:



14.31.3.4 StringToValue()

```

template<typename T >
bool AAX_CPercentDisplayDelegateDecorator< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
  
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

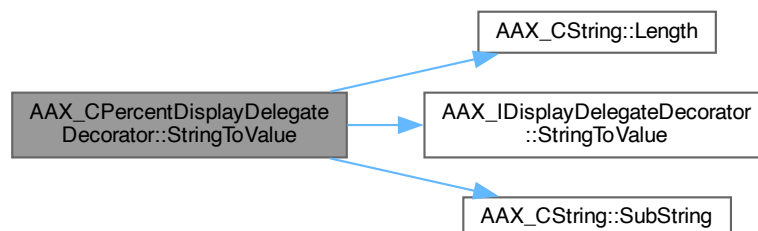
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Length\(\)](#), [AAX_IDisplayDelegateDecorator< T >::StringToValue\(\)](#), and [AAX_CString::SubString\(\)](#).

Here is the call graph for this function:



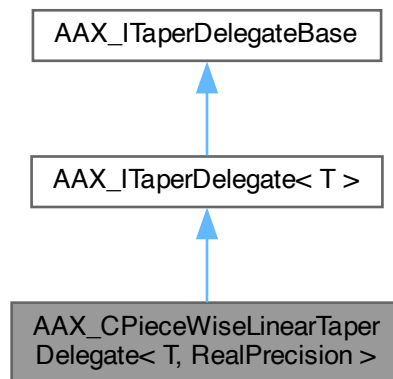
The documentation for this class was generated from the following file:

- [AAX_CPercentDisplayDelegateDecorator.h](#)

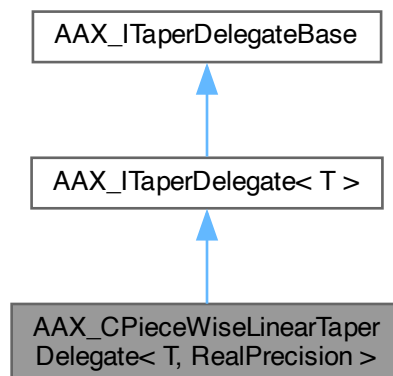
14.32 AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision > Class Template Reference

```
#include <AAX_CPieceWiseLinearTaperDelegate.h>
```

Inheritance diagram for AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >:



Collaboration diagram for AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >:



14.32.1 Description

```
template<typename T, int32_t RealPrecision = 100>
class AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >
```

A piece-wise linear taper conforming to [AAX_ITaperDelegate](#).

This taper spaces a parameter's real values in a piecewise linear fashion.

RealPrecision

In addition to its type templization, this taper includes a precision template parameter. RealPrecision is a multiplier that works in conjunction with the round() function to limit the precision of the real values provided by this taper. For example, if RealPrecision is 1000, it will round to the closest 0.001 when doing any sort of value conversion. If RealPrecision is 1, it will round to the nearest integer. If RealPrecision is 1000000, it will round to the nearest 0.000001. This is particularly useful for preventing things like 1.9999999 truncating down to 1 instead of rounding up to 2.

To accomplish this behavior, the taper multiplies its unrounded parameter values by RealPrecision, rounds the result to the nearest valid value, then divides RealPrecision back out.

Rounding will be disabled if RealPrecision is set to a value less than 1

Public Member Functions

- [AAX_CPieceWiseLinearTaperDelegate](#) (const double *normalizedValues, const T *realValues, int32_t numValues)
 - Constructs a Piece-wise Linear Taper with paired normalized and real values.*
- [AAX_CPieceWiseLinearTaperDelegate](#) (const [AAX_CPieceWiseLinearTaperDelegate](#) &other)
- [~AAX_CPieceWiseLinearTaperDelegate](#) ()
- [AAX_CPieceWiseLinearTaperDelegate](#)< T, RealPrecision > * [Clone](#) () const [AAX_OVERRIDE](#)
 - Constructs and returns a copy of the taper delegate.*
- T [GetMinimumValue](#) () const [AAX_OVERRIDE](#)
 - Returns the taper's minimum real value.*
- T [GetMaximumValue](#) () const [AAX_OVERRIDE](#)
 - Returns the taper's maximum real value.*
- T [ConstrainRealValue](#) (T value) const [AAX_OVERRIDE](#)
 - Applies a constraint to the value and returns the constrained value.*
- T [NormalizedToReal](#) (double normalizedValue) const [AAX_OVERRIDE](#)
 - Converts a normalized value to a real value.*
- double [RealToNormalized](#) (T realValue) const [AAX_OVERRIDE](#)
 - Normalizes a real parameter value.*
- virtual [AAX_ITaperDelegate](#) * [Clone](#) () const =0
 - Constructs and returns a copy of the taper delegate.*
- virtual T [GetMaximumValue](#) () const =0
 - Returns the taper's maximum real value.*
- virtual T [GetMinimumValue](#) () const =0
 - Returns the taper's minimum real value.*
- virtual T [ConstrainRealValue](#) (T value) const =0
 - Applies a constraint to the value and returns the constrained value.*
- virtual T [NormalizedToReal](#) (double normalizedValue) const =0
 - Converts a normalized value to a real value.*
- virtual double [RealToNormalized](#) (T realValue) const =0
 - Normalizes a real parameter value.*

Public Member Functions inherited from [AAX_ITaperDelegateBase](#)

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

Protected Member Functions

- T [Round](#) (double iValue) const

14.32.2 Constructor & Destructor Documentation

14.32.2.1 [AAX_CPieceWiseLinearTaperDelegate\(\)](#) [1/2]

```
template<typename T , int32_t RealPrecision>
AAX\_CPieceWiseLinearTaperDelegate< T, RealPrecision >::AAX\_CPieceWiseLinearTaperDelegate (
    const double * normalizedValues,
    const T * realValues,
    int32_t numValues )
```

Constructs a Piece-wise Linear Taper with paired normalized and real values.

Note

The parameter's default value should lie within the min to max range.

Parameters

in	<i>normalizedValues</i>	is an array of the normalized values in sorted order. (make sure to include the full normalized range, 0.0-1.0 inclusive)
in	<i>realValues</i>	is an array of the corresponding real values to the normalized values passed in.
in	<i>numValues</i>	is the number of values that have been passed in (i.e. the element length of the other input arrays)

14.32.2.2 [AAX_CPieceWiseLinearTaperDelegate\(\)](#) [2/2]

```
template<typename T , int32_t RealPrecision>
AAX\_CPieceWiseLinearTaperDelegate< T, RealPrecision >::AAX\_CPieceWiseLinearTaperDelegate (
    const AAX\_CPieceWiseLinearTaperDelegate< T, RealPrecision > & other )
```

14.32.2.3 [~AAX_CPieceWiseLinearTaperDelegate\(\)](#)

```
template<typename T , int32_t RealPrecision>
AAX\_CPieceWiseLinearTaperDelegate< T, RealPrecision >::~~AAX\_CPieceWiseLinearTaperDelegate
```


14.32.3 Member Function Documentation

14.32.3.1 Clone()

```
template<typename T , int32_t RealPrecision>
AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision > * AAX_CPieceWiseLinearTaperDelegate< T,
RealPrecision >::Clone ( ) const [virtual]
```

Constructs and returns a copy of the taper delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassTaperDelegate<T>* AAX_CSubclassTaperDelegate<T>::Clone() const
{
    return new AAX_CSubclassTaperDelegate(*this);
}
```

Implements [AAX_ITaperDelegate< T >](#).

14.32.3.2 GetMinimumValue()

```
template<typename T , int32_t RealPrecision = 100>
T AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >::GetMinimumValue ( ) const [inline],
[virtual]
```

Returns the taper's minimum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.32.3.3 GetMaximumValue()

```
template<typename T , int32_t RealPrecision = 100>
T AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >::GetMaximumValue ( ) const [inline],
[virtual]
```

Returns the taper's maximum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.32.3.4 ConstrainRealValue()

```
template<typename T , int32_t RealPrecision>
T AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >::ConstrainRealValue (
    T value ) const [virtual]
```

Applies a constraint to the value and returns the constrained value.

This method is useful if the taper requires a constraint beyond simple minimum and maximum real value limits.

Note

This is the function that should actually enforces the constraints in [NormalizeToReal\(\)](#) and [RealToNormalized\(\)](#).

Parameters

in	<i>value</i>	The unconstrained value
----	--------------	-------------------------

Implements [AAX_ITaperDelegate< T >](#).

14.32.3.5 NormalizedToReal()

```
template<typename T , int32_t RealPrecision>
T AAX\_CPieceWiseLinearTaperDelegate< T, RealPrecision >::NormalizedToReal (
    double normalizedValue ) const [virtual]
```

Converts a normalized value to a real value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [RealToNormalized\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>normalizedValue</i>	The normalized value that will be converted
----	------------------------	---

Implements [AAX_ITaperDelegate< T >](#).

14.32.3.6 RealToNormalized()

```
template<typename T , int32_t RealPrecision>
double AAX\_CPieceWiseLinearTaperDelegate< T, RealPrecision >::RealToNormalized (
    T realValue ) const [virtual]
```

Normalizes a real parameter value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [NormalizedToReal\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>realValue</i>	The real parameter value that will be normalized
----	------------------	--

Implements [AAX_ITaperDelegate< T >](#).

14.32.3.7 Round()

```
template<typename T , int32_t RealPrecision>
T AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >::Round (
    double iValue ) const [protected]
```

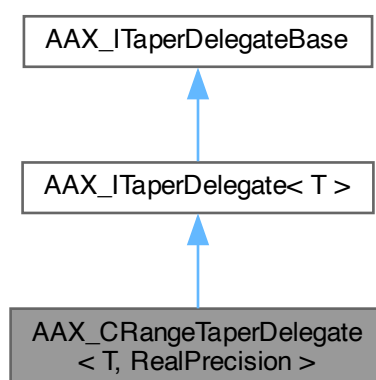
The documentation for this class was generated from the following file:

- [AAX_CPieceWiseLinearTaperDelegate.h](#)

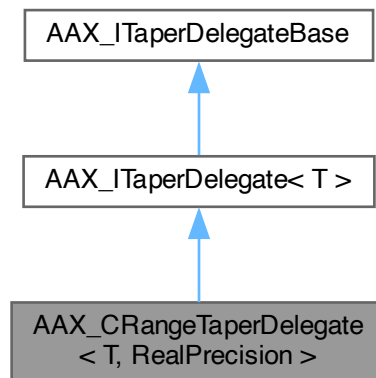
14.33 AAX_CRangeTaperDelegate< T, RealPrecision > Class Template Reference

```
#include <AAX_CRangeTaperDelegate.h>
```

Inheritance diagram for AAX_CRangeTaperDelegate< T, RealPrecision >:



Collaboration diagram for `AAX_CRangeTaperDelegate< T, RealPrecision >`:



14.33.1 Description

```
template<typename T, int32_t RealPrecision = 1000>
class AAX_CRangeTaperDelegate< T, RealPrecision >
```

A piecewise-linear taper conforming to [AAX_ITaperDelegate](#).

This taper spaces a parameter's real values between its minimum and maximum using a series of linear regions to create the full mapping between the parameter's real and normalized values.

Here is an example of how this taper can be used:

```
float rangePoints[] = { 0.0, 1.0, 100.0, 1000.0, 2000.0 };
double rangeSteps[] = { 0.1, 1.0, 10.0, 25.0 }; // number of steps per range: 10, 99, 90, 40
const long cNumRanges = sizeof(rangeSteps)/sizeof(rangeSteps[0]);

long numSteps = 0;
for (int i = 0; i < cNumRanges; i++)
{
    numSteps += (rangePoints[i+1] - rangePoints[i]) / rangeSteps[i];
}

AAX_CRangeTaperDelegate<float> nonLinearTaper(rangePoints, rangeSteps, cNumRanges);

float controlValue = 1.5;

double normalized = nonLinearTaper.RealToNormalized(controlValue);
float real = nonLinearTaper.NormalizedToReal(normalized);
```

RealPrecision

In addition to its type templization, this taper includes a precision template parameter. `RealPrecision` is a multiplier that works in conjunction with the `round()` function to limit the precision of the real values provided by this taper. For example, if `RealPrecision` is 1000, it will round to the closest 0.001 when doing any sort of value conversion. If `RealPrecision` is 1, it will round to the nearest integer. If `RealPrecision` is 1000000, it will round to the nearest 0.000001. This is particularly useful for preventing things like 1.9999999 truncating down to 1 instead of rounding up to 2.

To accomplish this behavior, the taper multiplies its unrounded parameter values by `RealPrecision`, rounds the result to the nearest valid value, then divides `RealPrecision` back out.

Rounding will be disabled if `RealPrecision` is set to a value less than 1

Public Member Functions

- [AAX_CRangeTaperDelegate](#) (T *range, double *rangesSteps, long numRanges, bool useSmartRounding=true)
Constructs a Range Taper with specified minimum and maximum values.
- [AAX_CRangeTaperDelegate](#) (const [AAX_CRangeTaperDelegate](#) &rhs)
- [AAX_CRangeTaperDelegate](#) & [operator=](#) ([AAX_CRangeTaperDelegate](#) &rhs)
- [AAX_CRangeTaperDelegate](#)< T, RealPrecision > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the taper delegate.
- T [GetMinimumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's minimum real value.
- T [GetMaximumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's maximum real value.
- T [ConstrainRealValue](#) (T value) const [AAX_OVERRIDE](#)
Applies a constraint to the value and returns the constrained value.
- T [NormalizedToReal](#) (double normalizedValue) const [AAX_OVERRIDE](#)
Converts a normalized value to a real value.
- double [RealToNormalized](#) (T realValue) const [AAX_OVERRIDE](#)
Normalizes a real parameter value.

- virtual [AAX_ITaperDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the taper delegate.
- virtual T [GetMaximumValue](#) () const =0
Returns the taper's maximum real value.
- virtual T [GetMinimumValue](#) () const =0
Returns the taper's minimum real value.
- virtual T [ConstrainRealValue](#) (T value) const =0
Applies a constraint to the value and returns the constrained value.
- virtual T [NormalizedToReal](#) (double normalizedValue) const =0
Converts a normalized value to a real value.
- virtual double [RealToNormalized](#) (T realValue) const =0
Normalizes a real parameter value.

Public Member Functions inherited from [AAX_ITaperDelegateBase](#)

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

Protected Member Functions

- T [Round](#) (double iValue) const
- T [SmartRound](#) (double value) const

14.33.2 Constructor & Destructor Documentation

14.33.2.1 AAX_CRangeTaperDelegate() [1/2]

```
template<typename T , int32_t RealPrecision>
AAX_CRangeTaperDelegate< T, RealPrecision >::AAX_CRangeTaperDelegate (
    T * range,
    double * rangesSteps,
    long numRanges,
    bool useSmartRounding = true )
```

Constructs a Range Taper with specified minimum and maximum values.

Note

The parameter's default value should lie within the min to max range.

Parameters

in	<i>range</i>	An array of range endpoints along the taper's mapping range
in	<i>rangesSteps</i>	Step values for each region in the taper's stepwise-linear map. No values in this array may be zero.
in	<i>numRanges</i>	The total number of linear regions in the taper's map
in	<i>useSmartRounding</i>	

Todo Document useSmartRounding parameter

14.33.2.2 AAX_CRangeTaperDelegate() [2/2]

```
template<typename T , int32_t RealPrecision>
AAX_CRangeTaperDelegate< T, RealPrecision >::AAX_CRangeTaperDelegate (
    const AAX_CRangeTaperDelegate< T, RealPrecision > & rhs )
```

14.33.3 Member Function Documentation

14.33.3.1 operator=()

```
template<typename T , int32_t RealPrecision>
AAX_CRangeTaperDelegate< T, RealPrecision > & AAX_CRangeTaperDelegate< T, RealPrecision >↔
::operator= (
    AAX_CRangeTaperDelegate< T, RealPrecision > & rhs )
```

14.33.3.2 Clone()

```
template<typename T , int32_t RealPrecision>
AAX_CRangeTaperDelegate< T, RealPrecision > * AAX_CRangeTaperDelegate< T, RealPrecision >::↵
Clone ( ) const [virtual]
```

Constructs and returns a copy of the taper delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassTaperDelegate<T>* AAX_CSubclassTaperDelegate<T>::Clone() const
{
    return new AAX_CSubclassTaperDelegate(*this);
}
```

Implements [AAX_ITaperDelegate< T >](#).

14.33.3.3 GetMinimumValue()

```
template<typename T , int32_t RealPrecision = 1000>
T AAX_CRangeTaperDelegate< T, RealPrecision >::GetMinimumValue ( ) const [inline], [virtual]
```

Returns the taper's minimum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.33.3.4 GetMaximumValue()

```
template<typename T , int32_t RealPrecision = 1000>
T AAX_CRangeTaperDelegate< T, RealPrecision >::GetMaximumValue ( ) const [inline], [virtual]
```

Returns the taper's maximum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.33.3.5 ConstrainRealValue()

```
template<typename T , int32_t RealPrecision>
T AAX_CRangeTaperDelegate< T, RealPrecision >::ConstrainRealValue (
    T value ) const [virtual]
```

Applies a constraint to the value and returns the constrained value.

This method is useful if the taper requires a constraint beyond simple minimum and maximum real value limits.

Note

This is the function that should actually enforces the constraints in [NormalizeToReal\(\)](#) and [RealToNormalized\(\)](#).

Parameters

in	<i>value</i>	The unconstrained value
----	--------------	-------------------------

Implements [AAX_ITaperDelegate< T >](#).

14.33.3.6 NormalizedToReal()

```
template<typename T , int32_t RealPrecision>
T AAX\_CRangeTaperDelegate< T, RealPrecision >::NormalizedToReal (
    double normalizedValue ) const [virtual]
```

Converts a normalized value to a real value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [RealToNormalized\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>normalizedValue</i>	The normalized value that will be converted
----	------------------------	---

Implements [AAX_ITaperDelegate< T >](#).

14.33.3.7 RealToNormalized()

```
template<typename T , int32_t RealPrecision>
double AAX\_CRangeTaperDelegate< T, RealPrecision >::RealToNormalized (
    T realValue ) const [virtual]
```

Normalizes a real parameter value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [NormalizedToReal\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>realValue</i>	The real parameter value that will be normalized
----	------------------	--

Implements [AAX_ITaperDelegate< T >](#).

14.33.3.8 Round()

```
template<typename T , int32_t RealPrecision>
T AAX_CRangeTaperDelegate< T, RealPrecision >::Round (
    double iValue ) const [protected]
```

14.33.3.9 SmartRound()

```
template<typename T , int32_t RealPrecision>
T AAX_CRangeTaperDelegate< T, RealPrecision >::SmartRound (
    double value ) const [protected]
```

Todo Document

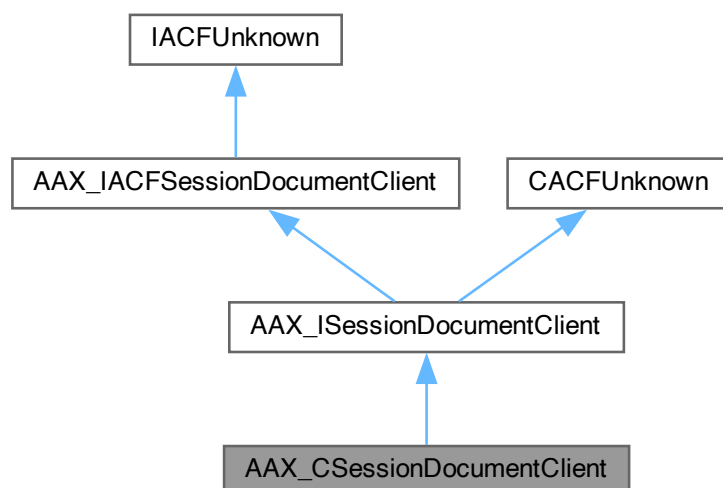
The documentation for this class was generated from the following file:

- [AAX_CRangeTaperDelegate.h](#)

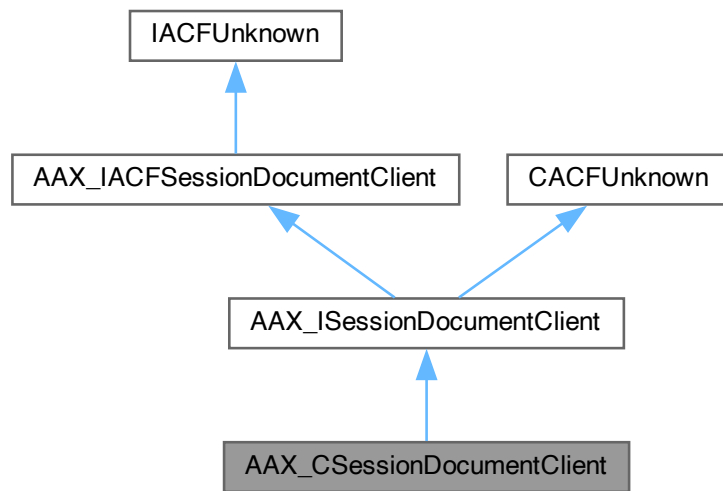
14.34 AAX_CSessionDocumentClient Class Reference

```
#include <AAX_CSessionDocumentClient.h>
```

Inheritance diagram for AAX_CSessionDocumentClient:



Collaboration diagram for AAX_CSessionDocumentClient:



14.34.1 Description

Default implementation of the [AAX_ISessionDocumentClient](#) interface.

Public Member Functions

- [AAX_CSessionDocumentClient](#) (void)
- [~AAX_CSessionDocumentClient](#) (void) [AAX_OVERRIDE](#)

Initialization and uninitialization

- [AAX_Result Initialize](#) ([IACFUnknown](#) *iUnknown) [AAX_OVERRIDE](#)
- [AAX_Result Uninitialize](#) (void) [AAX_OVERRIDE](#)

Session document access

- [AAX_Result SetSessionDocument](#) ([IACFUnknown](#) *iSessionDocument) [AAX_OVERRIDE](#)

AAX host and plug-in event notification

- [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#), const void *, uint32_t) [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_ISessionDocumentClient](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfIID](#) &riid
- [AAX_DELETE](#) ([AAX_ISessionDocumentClient](#) &operator=(const [AAX_ISessionDocumentClient](#) &))

Initialization and uninitialization**Session document access****AAX host and plug-in event notification****Public Member Functions inherited from [IACFUnknown](#)**

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Protected Member Functions**Session document change notifications**

- virtual [AAX_Result SessionDocumentWillChange](#) ()
The session document interface is about to be added, replaced, or removed.
- virtual [AAX_Result SessionDocumentChanged](#) ()
The session document interface has been added, replaced, or removed.

Private member accessors

- [AAX_IController](#) * [GetController](#) (void)
Retrieves a reference to the plug-in's controller interface.
- const [AAX_IController](#) * [GetController](#) (void) const
Retrieves a reference to the plug-in's controller interface.
- [AAX_IEffectParameters](#) * [GetEffectParameters](#) (void)
Retrieves a reference to the plug-in's data model interface.
- const [AAX_IEffectParameters](#) * [GetEffectParameters](#) (void) const
Retrieves a reference to the plug-in's data model interface.
- std::shared_ptr< [AAX_ISessionDocument](#) > [GetSessionDocument](#) (void)
Retrieves a reference to the session document interface.
- std::shared_ptr< const [AAX_ISessionDocument](#) > [GetSessionDocument](#) (void) const
Retrieves a reference to the session document interface.

Additional Inherited Members**Public Attributes inherited from [AAX_ISessionDocumentClient](#)**

- void **ppvObjOut [override](#)

14.34.2 Constructor & Destructor Documentation

14.34.2.1 AAX_CSessionDocumentClient()

```
AAX_CSessionDocumentClient::AAX_CSessionDocumentClient (
    void )
```

14.34.2.2 ~AAX_CSessionDocumentClient()

```
AAX_CSessionDocumentClient::~~AAX_CSessionDocumentClient (
    void )
```

14.34.3 Member Function Documentation

14.34.3.1 Initialize()

```
AAX_Result AAX_CSessionDocumentClient::Initialize (
    IACFUnknown * iUnknown ) [virtual]
```

Implements [AAX_IACFSessionDocumentClient](#).

14.34.3.2 Uninitialize()

```
AAX_Result AAX_CSessionDocumentClient::Uninitialize (
    void ) [virtual]
```

Implements [AAX_IACFSessionDocumentClient](#).

14.34.3.3 SetSessionDocument()

```
AAX_Result AAX_CSessionDocumentClient::SetSessionDocument (
    IACFUnknown * iSessionDocument ) [virtual]
```

Sets or removes a session document.

Parameters

in	<i>iSessionDocument</i>	Interface supporting at least AAX_IACFSessionDocument , or <code>nullptr</code> to indicate that any session document that is currently held should be released.
----	-------------------------	--

Implements [AAX_IACFSessionDocumentClient](#).

14.34.3.4 NotificationReceived()

```
AAX_Result AAX_CSessionDocumentClient::NotificationReceived (
    AAX_CTypeID ,
    const void * ,
    uint32_t ) [inline], [virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- Different notifications are sent to different objects within a plug-in. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

A plug-in may also dispatch custom notifications using [AAX_IController::SendNotification\(\)](#). Custom notifications will be posted back to the plug-in's other objects which support a [NotificationReceived\(\)](#) method (e.g. the data model).

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implements [AAX_IACFSessionDocumentClient](#).

References [AAX_SUCCESS](#).

14.34.3.5 SessionDocumentWillChange()

```
virtual AAX_Result AAX_CSessionDocumentClient::SessionDocumentWillChange ( ) [inline], [protected],
[virtual]
```

The session document interface is about to be added, replaced, or removed.

Custom implementations should stop using the current session document interface, which is about to become invalid.

References [AAX_SUCCESS](#).

14.34.3.6 SessionDocumentChanged()

```
virtual AAX\_Result AAX_CSessionDocumentClient::SessionDocumentChanged ( ) [inline], [protected],  
[virtual]
```

The session document interface has been added, replaced, or removed.

Custom implementations should update local references to the session document interface.

References [AAX_SUCCESS](#).

14.34.3.7 GetController() [1/2]

```
AAX\_IController * AAX_CSessionDocumentClient::GetController (   
    void ) [protected]
```

Retrieves a reference to the plug-in's controller interface.

14.34.3.8 GetController() [2/2]

```
const AAX\_IController * AAX_CSessionDocumentClient::GetController (   
    void ) const [protected]
```

Retrieves a reference to the plug-in's controller interface.

14.34.3.9 GetEffectParameters() [1/2]

```
AAX\_IEffectParameters * AAX_CSessionDocumentClient::GetEffectParameters (   
    void ) [protected]
```

Retrieves a reference to the plug-in's data model interface.

14.34.3.10 GetEffectParameters() [2/2]

```
const AAX_IEffectParameters * AAX_CSessionDocumentClient::GetEffectParameters (
    void ) const [protected]
```

Retrieves a reference to the plug-in's data model interface.

14.34.3.11 GetSessionDocument() [1/2]

```
std::shared_ptr< AAX_ISessionDocument > AAX_CSessionDocumentClient::GetSessionDocument (
    void ) [protected]
```

Retrieves a reference to the session document interface.

14.34.3.12 GetSessionDocument() [2/2]

```
std::shared_ptr< const AAX_ISessionDocument > AAX_CSessionDocumentClient::GetSessionDocument (
    void ) const [protected]
```

Retrieves a reference to the session document interface.

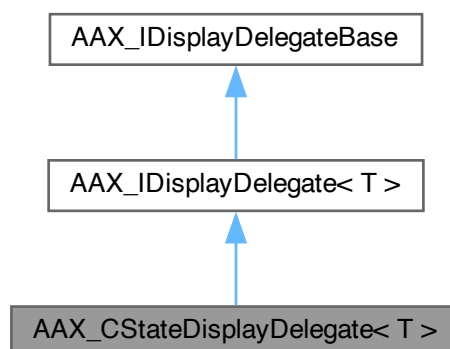
The documentation for this class was generated from the following file:

- [AAX_CSessionDocumentClient.h](#)

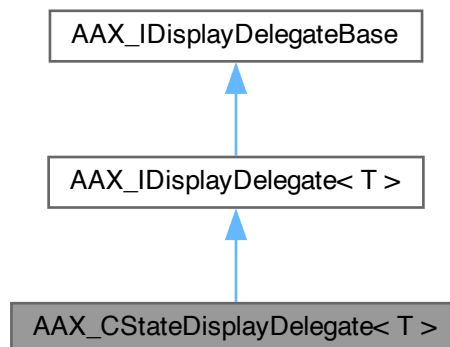
14.35 AAX_CStateDisplayDelegate< T > Class Template Reference

```
#include <AAX_CStateDisplayDelegate.h>
```

Inheritance diagram for AAX_CStateDisplayDelegate< T >:



Collaboration diagram for `AAX_CStateDisplayDelegate< T >`:



14.35.1 Description

```
template<typename T>
class AAX_CStateDisplayDelegate< T >
```

A generic display format conforming to [AAX_IDisplayDelegate](#).

This display delegate is similar to [AAX_CNumberDisplayDelegate](#), but does not include precision or spacing templizations.

Public Member Functions

- [AAX_CStateDisplayDelegate](#) (const char *iStateStrings[], T iMinState=0)
Constructor taking a vector of C strings.
- [AAX_CStateDisplayDelegate](#) (int32_t inNumStates, const char *iStateStrings[], T iMinState=0)
Constructor taking a vector of C strings.
- [AAX_CStateDisplayDelegate](#) (const std::vector< [AAX_IString](#) * > &iStateStrings, T iMinState=0)
Constructor taking a vector of [AAX_IString](#) objects.
- [AAX_CStateDisplayDelegate](#) (const [AAX_CStateDisplayDelegate](#) &other)
- [AAX_IDisplayDelegate< T > * Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- void [AddShortenedStrings](#) (const char *iStateStrings[], int iLength)
- bool [Compare](#) (const [AAX_CString](#) &valueString, const [AAX_CString](#) &stateString) const

- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.35.2 Constructor & Destructor Documentation

14.35.2.1 AAX_CStateDisplayDelegate() [1/4]

```
template<typename T >
AAX_CStateDisplayDelegate< T >::AAX_CStateDisplayDelegate (
    const char * iStateStrings[],
    T iMinState = 0 ) [explicit]
```

Constructor taking a vector of C strings.

Each state name will be copied into the display delegate; the C strings may be disposed after construction.

Note

`iStateStrings` must be NULL-terminated

14.35.2.2 AAX_CStateDisplayDelegate() [2/4]

```
template<typename T >
AAX_CStateDisplayDelegate< T >::AAX_CStateDisplayDelegate (
    int32_t inNumStates,
    const char * iStateStrings[],
    T iMinState = 0 ) [explicit]
```

Constructor taking a vector of C strings.

Each state name will be copied into the display delegate; the C strings may be disposed after construction.

State strings will be copied into the display delegate until either a NULL pointer is encountered or `inNumStates` strings have been copied

14.35.2.3 AAX_CStateDisplayDelegate() [3/4]

```
template<typename T >
AAX_CStateDisplayDelegate< T >::AAX_CStateDisplayDelegate (
    const std::vector< AAX_IString * > & iStateStrings,
    T iMinState = 0 ) [explicit]
```

Constructor taking a vector of [AAX_IString](#) objects.

Each [AAX_IString](#) will be copied into the display delegate and may be disposed after construction. The [AAX_IString](#) will not be mutated.

14.35.2.4 AAX_CStateDisplayDelegate() [4/4]

```
template<typename T >
AAX_CStateDisplayDelegate< T >::AAX_CStateDisplayDelegate (
    const AAX_CStateDisplayDelegate< T > & other )
```

14.35.3 Member Function Documentation

14.35.3.1 Clone()

```
template<typename T >
AAX_IDisplayDelegate< T > * AAX_CStateDisplayDelegate< T >::Clone ( ) const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>*    AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.35.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX_CStateDisplayDelegate< T >::ValueToString (
    T value,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.35.3.3 ValueToString() [2/2]

```
template<typename T >
bool AAX_CStateDisplayDelegate< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.35.3.4 StringToValue()

```
template<typename T >
bool AAX_CStateDisplayDelegate< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.35.3.5 AddShortenedStrings()

```
template<typename T >
void AAX_CStateDisplayDelegate< T >::AddShortenedStrings (
    const char * iStateStrings[],
    int iLength )
```

14.35.3.6 Compare()

```
template<typename T >
bool AAX_CStateDisplayDelegate< T >::Compare (
    const AAX_CString & valueString,
    const AAX_CString & stateString ) const
```

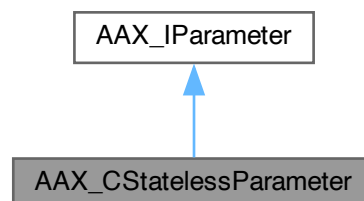
The documentation for this class was generated from the following file:

- [AAX_CStateDisplayDelegate.h](#)

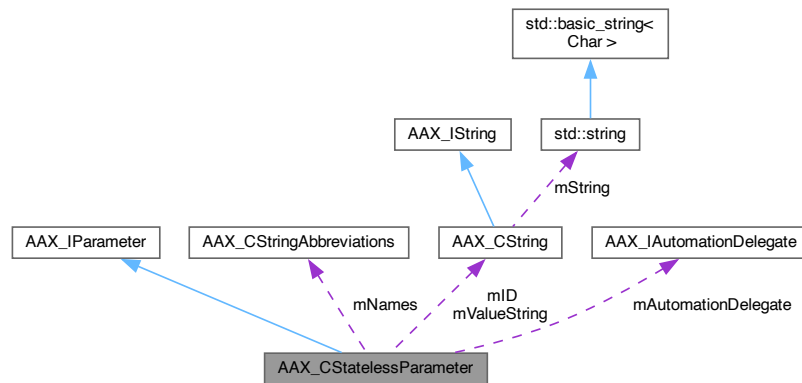
14.36 AAX_CStatelessParameter Class Reference

```
#include <AAX_CParameter.h>
```

Inheritance diagram for AAX_CStatelessParameter:



Collaboration diagram for AAX_CStatelessParameter:



14.36.1 Description

A stateless parameter implementation.

This can be useful for mapping event triggers to control surface buttons or to GUI switches.

Public Member Functions

- [AAX_CStatelessParameter](#) ([AAX_CParamID](#) identifier, const [AAX_IString](#) &name, const [AAX_IString](#) &in↔ValueString)
- [AAX_CStatelessParameter](#) (const [AAX_IString](#) &identifier, const [AAX_IString](#) &name, const [AAX_IString](#) &inValueString)
- [AAX_DEFAULT_DTOR_OVERRIDE](#) ([AAX_CStatelessParameter](#))
- [AAX_IPParameterValue * CloneValue](#) () const [AAX_OVERRIDE](#)
Clone the parameter's value to a new [AAX_IPParameterValue](#) object.

Identification methods

- [AAX_CParamID Identifier](#) () const [AAX_OVERRIDE](#)
Returns the parameter's unique identifier.
- void [SetName](#) (const [AAX_CString](#) &name) [AAX_OVERRIDE](#)
Sets the parameter's display name.
- const [AAX_CString & Name](#) () const [AAX_OVERRIDE](#)
Returns the parameter's display name.
- void [AddShortenedName](#) (const [AAX_CString](#) &name) [AAX_OVERRIDE](#)
Sets the parameter's shortened display name.
- const [AAX_CString & ShortenedName](#) (int32_t iNumCharacters) const [AAX_OVERRIDE](#)
Returns the parameter's shortened display name.
- void [ClearShortenedNames](#) () [AAX_OVERRIDE](#)
Clears the internal list of shortened display names.

Automation methods

- bool [Automatable](#) () const [AAX_OVERRIDE](#)

- Returns true if the parameter is automatable, false if it is not.*
- void [SetAutomationDelegate](#) ([AAX_IAutomationDelegate](#) *iAutomationDelegate) [AAX_OVERRIDE](#)
Sets the automation delegate (if one is required)
- void [Touch](#) () [AAX_OVERRIDE](#)
Signals the automation system that a control has been touched.
- void [Release](#) () [AAX_OVERRIDE](#)
Signals the automation system that a control has been released.

Taper methods

- void [SetNormalizedValue](#) (double) [AAX_OVERRIDE](#)
Sets a parameter value using it's normalized representation.
- double [GetNormalizedValue](#) () const [AAX_OVERRIDE](#)
Returns the normalized representation of the parameter's current real value.
- void [SetNormalizedDefaultValue](#) (double) [AAX_OVERRIDE](#)
Sets the parameter's default value using its normalized representation.
- double [GetNormalizedDefaultValue](#) () const [AAX_OVERRIDE](#)
Returns the normalized representation of the parameter's real default value.
- void [SetToDefaultValue](#) () [AAX_OVERRIDE](#)
Restores the state of this parameter to its default value.
- void [SetNumberOfSteps](#) (uint32_t) [AAX_OVERRIDE](#)
Sets the number of discrete steps for this parameter.
- uint32_t [GetNumberOfSteps](#) () const [AAX_OVERRIDE](#)
Returns the number of discrete steps used by the parameter.
- uint32_t [GetStepValue](#) () const [AAX_OVERRIDE](#)
Returns the current step for the current value of the parameter.
- double [GetNormalizedValueFromStep](#) (uint32_t) const [AAX_OVERRIDE](#)
Returns the normalized value for a given step.
- uint32_t [GetStepValueFromNormalizedValue](#) (double) const [AAX_OVERRIDE](#)
Returns the step value for a normalized value of the parameter.
- void [SetStepValue](#) (uint32_t) [AAX_OVERRIDE](#)
Returns the current step for the current value of the parameter.

Display methods

This functionality is most often used by GUIs, but can also be useful for state serialization.

- bool [GetValueString](#) ([AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Serializes the parameter value into a string.
- bool [GetValueString](#) (int32_t, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Serializes the parameter value into a string, size hint included.
- bool [GetNormalizedValueFromBool](#) (bool, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a bool to a normalized parameter value.
- bool [GetNormalizedValueFromInt32](#) (int32_t, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts an integer to a normalized parameter value.
- bool [GetNormalizedValueFromFloat](#) (float, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a float to a normalized parameter value.
- bool [GetNormalizedValueFromDouble](#) (double, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a double to a normalized parameter value.
- bool [GetNormalizedValueFromString](#) (const [AAX_CString](#) &, double *normalizedValue) const [AAX_OVERRIDE](#)
Converts a given string to a normalized parameter value.
- bool [GetBoolFromNormalizedValue](#) (double, bool *value) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to a bool representing the corresponding real value.
- bool [GetInt32FromNormalizedValue](#) (double, int32_t *) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to an integer representing the corresponding real value.
- bool [GetFloatFromNormalizedValue](#) (double, float *) const [AAX_OVERRIDE](#)
Converts a normalized parameter value to a float representing the corresponding real value.
- bool [GetDoubleFromNormalizedValue](#) (double, double *) const [AAX_OVERRIDE](#)

- *Converts a normalized parameter value to a double representing the corresponding real value.*
 bool GetStringFromNormalizedValue (double, AAX_CString &valueString) const AAX_OVERRIDE
- *Converts a normalized parameter value to a string representing the corresponding real value.*
 bool GetStringFromNormalizedValue (double normalizedValue, int32_t, AAX_CString &valueString) const AAX_OVERRIDE
- *Converts a normalized parameter value to a string representing the corresponding real, size hint included. value.*
 bool SetValueFromString (const AAX_CString &newValueString) AAX_OVERRIDE
- *Converts a string to a real parameter value and sets the parameter to this value.*

Typed accessors

- bool GetValueAsBool (bool *value) const AAX_OVERRIDE
Retrieves the parameter's value as a bool.
- bool GetValueAsInt32 (int32_t *) const AAX_OVERRIDE
Retrieves the parameter's value as an int32_t.
- bool GetValueAsFloat (float *) const AAX_OVERRIDE
Retrieves the parameter's value as a float.
- bool GetValueAsDouble (double *) const AAX_OVERRIDE
Retrieves the parameter's value as a double.
- bool GetValueAsString (AAX_IString *) const AAX_OVERRIDE
Retrieves the parameter's value as a string.
- bool SetValueWithBool (bool) AAX_OVERRIDE
Sets the parameter's value as a bool.
- bool SetValueWithInt32 (int32_t) AAX_OVERRIDE
Sets the parameter's value as an int32_t.
- bool SetValueWithFloat (float) AAX_OVERRIDE
Sets the parameter's value as a float.
- bool SetValueWithDouble (double) AAX_OVERRIDE
Sets the parameter's value as a double.
- bool SetValueWithString (const AAX_IString &value) AAX_OVERRIDE
Sets the parameter's value as a string.
- void SetType (AAX_EParameterType) AAX_OVERRIDE
Sets the type of this parameter.
- AAX_EParameterType GetType () const AAX_OVERRIDE
Returns the type of this parameter as an AAX_EParameterType.
- void SetOrientation (AAX_EParameterOrientation) AAX_OVERRIDE
Sets the orientation of this parameter.
- AAX_EParameterOrientation GetOrientation () const AAX_OVERRIDE
Returns the orientation of this parameter.
- void SetTaperDelegate (AAX_ITaperDelegateBase &, bool) AAX_OVERRIDE
Sets the parameter's taper delegate.
- void SetDisplayDelegate (AAX_IDisplayDelegateBase &) AAX_OVERRIDE
Sets the parameter's display delegate.

Public Member Functions inherited from AAX_IParameter

- virtual ~AAX_IParameter ()
Virtual destructor.
- virtual AAX_IParameterValue * CloneValue () const =0
Clone the parameter's value to a new AAX_IParameterValue object.

Host interface methods

- AAX_CStringAbbreviations mNames
- AAX_CString mID
- AAX_IAutomationDelegate * mAutomationDelegate
- AAX_CString mValueString
- void UpdateNormalizedValue (double) AAX_OVERRIDE
Sets the parameter's state given a normalized value.

14.36.2 Constructor & Destructor Documentation

14.36.2.1 AAX_CStatelessParameter() [1/2]

```
AAX_CStatelessParameter::AAX_CStatelessParameter (
    AAX_CParamID identifier,
    const AAX_IString & name,
    const AAX_IString & inValueString ) [inline]
```

14.36.2.2 AAX_CStatelessParameter() [2/2]

```
AAX_CStatelessParameter::AAX_CStatelessParameter (
    const AAX_IString & identifier,
    const AAX_IString & name,
    const AAX_IString & inValueString ) [inline]
```

14.36.3 Member Function Documentation

14.36.3.1 AAX_DEFAULT_DTOR_OVERRIDE()

```
AAX_CStatelessParameter::AAX_DEFAULT_DTOR_OVERRIDE (
    AAX_CStatelessParameter )
```

14.36.3.2 CloneValue()

```
AAX_IParameterValue * AAX_CStatelessParameter::CloneValue ( ) const [inline], [virtual]
```

Clone the parameter's value to a new [AAX_IParameterValue](#) object.

The returned object is independent from the [AAX_IParameter](#). For example, changing the state of the returned object will not result in a change to the original [AAX_IParameter](#).

Implements [AAX_IParameter](#).

14.36.3.3 Identifier()

`AAX_CParamID AAX_CStatelessParameter::Identifier () const [inline], [virtual]`

Returns the parameter's unique identifier.

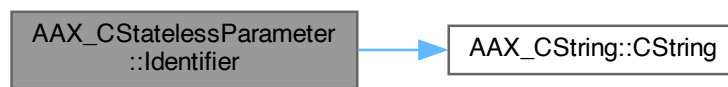
This unique ID is used by the [Parameter Manager](#) and by outside applications to uniquely identify and target control messages. This value may not be changed after the parameter has been constructed.

Implements [AAX_IParameter](#).

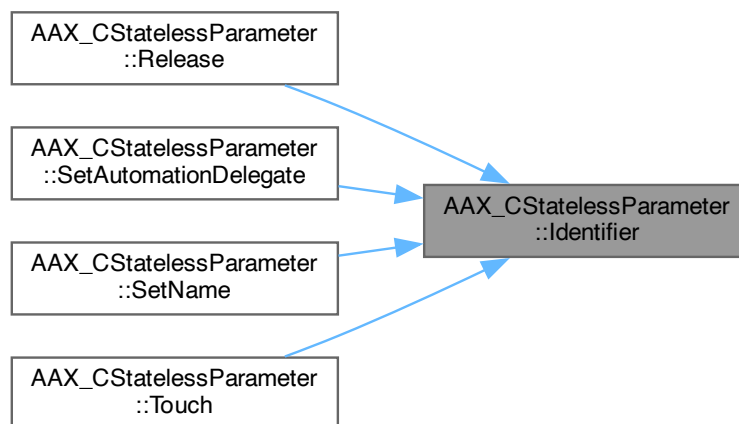
References [AAX_CString::CString\(\)](#), and [mID](#).

Referenced by [Release\(\)](#), [SetAutomationDelegate\(\)](#), [SetName\(\)](#), and [Touch\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.36.3.4 SetName()

`void AAX_CStatelessParameter::SetName (const AAX_CString & name) [inline], [virtual]`

Sets the parameter's display name.

This name is used for display only, it is not used for indexing or identifying the parameter. This name may be changed after the parameter has been created, but display name changes may not be recognized by all AAX hosts.

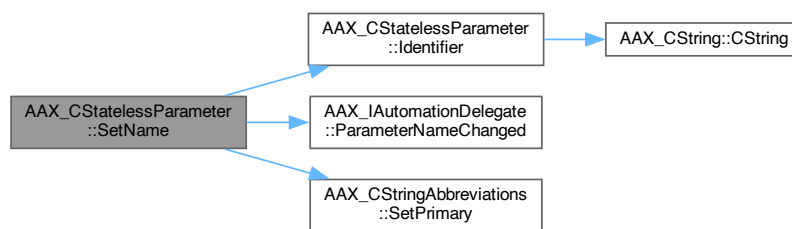
Parameters

in	name	Display name that will be assigned to the parameter
----	------	---

Implements [AAX_IParameter](#).

References [Identifier\(\)](#), [mAutomationDelegate](#), [mNames](#), [AAX_IAutomationDelegate::ParameterNameChanged\(\)](#), and [AAX_CStringAbbreviations::SetPrimary\(\)](#).

Here is the call graph for this function:



14.36.3.5 Name()

```
const AAX\_CString & AAX_CStatelessParameter::Name ( ) const [inline], [virtual]
```

Returns the parameter's display name.

Note

This method returns a const reference in order to prevent a string copy. Do not cast away the const to change this value.

Implements [AAX_IParameter](#).

References [mNames](#), and [AAX_CStringAbbreviations::Primary\(\)](#).

Here is the call graph for this function:



14.36.3.6 AddShortenedName()

```
void AAX_CStatelessParameter::AddShortenedName (
    const AAX_CString & name ) [inline], [virtual]
```

Sets the parameter's shortened display name.

This name is used for display only, it is not used for indexing or identifying the parameter. These names show up when the host asks for shorter length parameter names for display on Control Surfaces or other string length constrained situations.

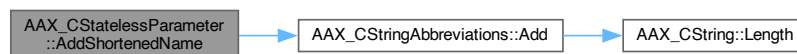
Parameters

in	<i>name</i>	Shortened display names that will be assigned to the parameter
----	-------------	--

Implements [AAX_IParameter](#).

References [AAX_CStringAbbreviations::Add\(\)](#), and [mNames](#).

Here is the call graph for this function:



14.36.3.7 ShortenedName()

```
const AAX_CString & AAX_CStatelessParameter::ShortenedName (
    int32_t iNumCharacters ) const [inline], [virtual]
```

Returns the parameter's shortened display name.

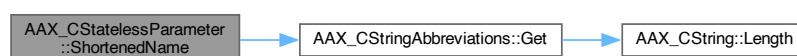
Note

This method returns a const reference in order to prevent a string copy. Do not cast away the const to change this value.

Implements [AAX_IParameter](#).

References [AAX_CStringAbbreviations::Get\(\)](#), and [mNames](#).

Here is the call graph for this function:



14.36.3.8 ClearShortenedNames()

```
void AAX_CStatelessParameter::ClearShortenedNames ( ) [inline], [virtual]
```

Clears the internal list of shortened display names.

Implements [AAX_IParameter](#).

References [AAX_CStringAbbreviations::Clear\(\)](#), and [mNames](#).

Here is the call graph for this function:



14.36.3.9 Automatable()

```
bool AAX_CStatelessParameter::Automatable ( ) const [inline], [virtual]
```

Returns true if the parameter is automatable, false if it is not.

Note

Subclasses that return true in this method must support host-based automation.

Implements [AAX_IParameter](#).

14.36.3.10 SetAutomationDelegate()

```
void AAX_CStatelessParameter::SetAutomationDelegate (
    AAX\_IAutomationDelegate * iAutomationDelegate ) [inline], [virtual]
```

Sets the automation delegate (if one is required)

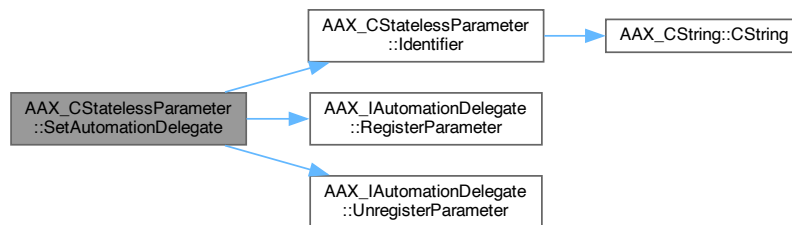
Parameters

in	<i>iAutomationDelegate</i>	A reference to the parameter manager's automation delegate interface
----	----------------------------	--

Implements [AAX_IParameter](#).

References [Identifier\(\)](#), [mAutomationDelegate](#), [AAX_IAutomationDelegate::RegisterParameter\(\)](#), and [AAX_IAutomationDelegate::UnregisterParameter\(\)](#).

Here is the call graph for this function:



14.36.3.11 Touch()

```
void AAX_CStatelessParameter::Touch ( ) [inline], [virtual]
```

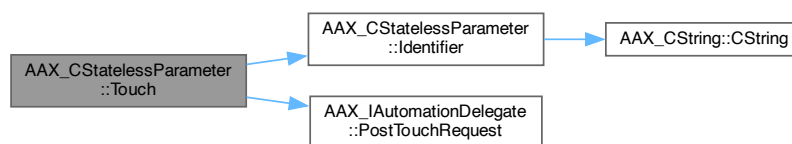
Signals the automation system that a control has been touched.

Call this method in response to GUI events that begin editing, such as a mouse down. After this method has been called you are free to call [SetNormalizedValue\(\)](#) as much as you need, e.g. in order to respond to subsequent mouse moved events. Call [Release\(\)](#) to free the parameter for updates from other controls.

Implements [AAX_IParameter](#).

References [Identifier\(\)](#), [mAutomationDelegate](#), and [AAX_IAutomationDelegate::PostTouchRequest\(\)](#).

Here is the call graph for this function:



14.36.3.12 Release()

```
void AAX_CStatelessParameter::Release ( ) [inline], [virtual]
```

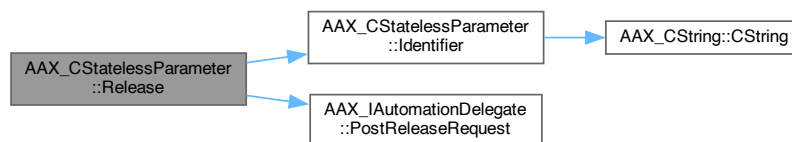
Signals the automation system that a control has been released.

Call this method in response to GUI events that complete editing, such as a mouse up. Once this method has been called you should not call [SetNormalizedValue\(\)](#) again until after the next call to [Touch\(\)](#).

Implements [AAX_IParameter](#).

References [Identifier\(\)](#), [mAutomationDelegate](#), and [AAX_IAutomationDelegate::PostReleaseRequest\(\)](#).

Here is the call graph for this function:



14.36.3.13 SetNormalizedValue()

```
void AAX_CStatelessParameter::SetNormalizedValue (
    double newNormalizedValue ) [inline], [virtual]
```

Sets a parameter value using it's normalized representation.

For more information regarding normalized values, see [Parameter Manager](#)

Parameters

in	<i>newNormalizedValue</i>	New value (normalized) to which the parameter will be set
----	---------------------------	---

Implements [AAX_IParameter](#).

14.36.3.14 GetNormalizedValue()

```
double AAX_CStatelessParameter::GetNormalizedValue ( ) const [inline], [virtual]
```

Returns the normalized representation of the parameter's current real value.

Implements [AAX_IParameter](#).

14.36.3.15 SetNormalizedDefaultValue()

```
void AAX_CStatelessParameter::SetNormalizedDefaultValue (
    double normalizedDefault ) [inline], [virtual]
```

Sets the parameter's default value using its normalized representation.

Implements [AAX_IParameter](#).

14.36.3.16 GetNormalizedDefaultValue()

```
double AAX_CStatelessParameter::GetNormalizedDefaultValue ( ) const [inline], [virtual]
```

Returns the normalized representation of the parameter's real default value.

Implements [AAX_IParameter](#).

14.36.3.17 SetToDefaultValue()

```
void AAX_CStatelessParameter::SetToDefaultValue ( ) [inline], [virtual]
```

Restores the state of this parameter to its default value.

Implements [AAX_IParameter](#).

14.36.3.18 SetNumberOfSteps()

```
void AAX_CStatelessParameter::SetNumberOfSteps (
    uint32_t numSteps ) [inline], [virtual]
```

Sets the number of discrete steps for this parameter.

Stepped parameter values are useful for discrete parameters and for "jumping" events such as mouse wheels, page up/down, etc. The parameter's step size is used to specify the coarseness of those changes.

Note

`numSteps` MUST be greater than zero. All other values may be considered an error by the host.

Parameters

<code>in</code>	<code><i>numSteps</i></code>	The number of steps that the parameter will use
-----------------	------------------------------	---

Implements [AAX_IParameter](#).

14.36.3.19 GetNumberOfSteps()

```
uint32_t AAX_CStatelessParameter::GetNumberOfSteps ( ) const [inline], [virtual]
```

Returns the number of discrete steps used by the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.36.3.20 GetStepValue()

```
uint32_t AAX_CStatelessParameter::GetStepValue ( ) const [inline], [virtual]
```

Returns the current step for the current value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.36.3.21 GetNormalizedValueFromStep()

```
double AAX_CStatelessParameter::GetNormalizedValueFromStep (
    uint32_t iStep ) const [inline], [virtual]
```

Returns the normalized value for a given step.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.36.3.22 GetStepValueFromNormalizedValue()

```
uint32_t AAX_CStatelessParameter::GetStepValueFromNormalizedValue (
    double normalizedValue ) const [inline], [virtual]
```

Returns the step value for a normalized value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.36.3.23 SetStepValue()

```
void AAX_CStatelessParameter::SetStepValue (
    uint32_t iStep ) [inline], [virtual]
```

Returns the current step for the current value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implements [AAX_IParameter](#).

14.36.3.24 GetValueString() [1/2]

```
bool AAX_CStatelessParameter::GetValueString (
    AAX_CString * valueString ) const [inline], [virtual]
```

Serializes the parameter value into a string.

Parameters

out	<i>valueString</i>	A string representing the parameter's real value
-----	--------------------	--

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

References [mValueString](#).

14.36.3.25 GetValueString() [2/2]

```
bool AAX_CStatelessParameter::GetValueString (
    int32_t iMaxNumChars,
    AAX_CString * valueString ) const [inline], [virtual]
```

Serializes the parameter value into a string, size hint included.

Parameters

in	<i>iMaxNumChars</i>	A size hint for the size of the string being requested. Useful for control surfaces and other limited area text fields. (make sure that size of desired string also has room for null termination)
out	<i>valueString</i>	A string representing the parameter's real value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

References [GetValueString\(\)](#).

Referenced by [GetValueString\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.36.3.26 GetNormalizedValueFromBool()

```

bool AAX_CStatelessParameter::GetNormalizedValueFromBool (
    bool value,
    double * normalizedValue ) const [inline], [virtual]
  
```

Converts a bool to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with <i>value</i>

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.27 GetNormalizedValueFromInt32()

```
bool AAX_CStatelessParameter::GetNormalizedValueFromInt32 (
    int32_t value,
    double * normalizedValue ) const [inline], [virtual]
```

Converts an integer to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.28 GetNormalizedValueFromFloat()

```
bool AAX_CStatelessParameter::GetNormalizedValueFromFloat (
    float value,
    double * normalizedValue ) const [inline], [virtual]
```

Converts a float to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.29 GetNormalizedValueFromDouble()

```
bool AAX_CStatelessParameter::GetNormalizedValueFromDouble (
    double value,
    double * normalizedValue ) const [inline], [virtual]
```

Converts a double to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.30 GetNormalizedValueFromString()

```
bool AAX_CStatelessParameter::GetNormalizedValueFromString (
    const AAX\_CString & valueString,
    double * normalizedValue ) const [inline], [virtual]
```

Converts a given string to a normalized parameter value.

Parameters

in	<i>valueString</i>	A string representing a possible real value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.31 GetBoolFromNormalizedValue()

```
bool AAX_CStatelessParameter::GetBoolFromNormalizedValue (
    double normalizedValue,
    bool * value ) const [inline], [virtual]
```

Converts a normalized parameter value to a bool representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.32 GetInt32FromNormalizedValue()

```
bool AAX_CStatelessParameter::GetInt32FromNormalizedValue (
    double normalizedValue,
    int32_t * value ) const [inline], [virtual]
```

Converts a normalized parameter value to an integer representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to int32_t was successful
<i>false</i>	The conversion to int32_t was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.33 GetFloatFromNormalizedValue()

```
bool AAX_CStatelessParameter::GetFloatFromNormalizedValue (
    double normalizedValue,
    float * value ) const [inline], [virtual]
```

Converts a normalized parameter value to a float representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.34 GetDoubleFromNormalizedValue()

```
bool AAX_CStatelessParameter::GetDoubleFromNormalizedValue (
    double normalizedValue,
    double * value ) const [inline], [virtual]
```

Converts a normalized parameter value to a double representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to double was successful
<i>false</i>	The conversion to double was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.35 GetStringFromNormalizedValue() [1/2]

```
bool AAX_CStatelessParameter::GetStringFromNormalizedValue (
    double normalizedValue,
    AAX_CString & valueString ) const [inline], [virtual]
```

Converts a normalized parameter value to a string representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	A normalized parameter value
out	<i>valueString</i>	A string representing the parameter value associated with normalizedValue

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

References [mValueString](#).

14.36.3.36 GetStringFromNormalizedValue() [2/2]

```
bool AAX_CStatelessParameter::GetStringFromNormalizedValue (
    double normalizedValue,
    int32_t iMaxNumChars,
    AAX_CString & valueString ) const [inline], [virtual]
```

Converts a normalized parameter value to a string representing the corresponding real, size hint included. value.

Parameters

in	<i>normalizedValue</i>	A normalized parameter value
in	<i>iMaxNumChars</i>	A size hint for the size of the string being requested. Useful for control surfaces and other limited area text fields. (make sure that size of desired string also has room for null termination)
out	<i>valueString</i>	A string representing the parameter value associated with normalizedValue

Return values

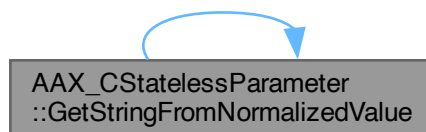
<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

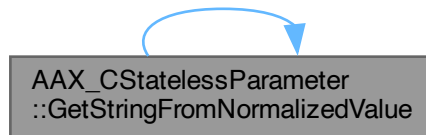
References [GetStringFromNormalizedValue\(\)](#).

Referenced by [GetStringFromNormalizedValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.36.3.37 SetValueFromString()

```
bool AAX_CStatelessParameter::SetValueFromString (
    const AAX_CString & newValueString ) [inline], [virtual]
```

Converts a string to a real parameter value and sets the parameter to this value.

Parameters

in	<i>newValueString</i>	A string representing the parameter's new real value
----	-----------------------	--

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IParameter](#).

References [mValueString](#).

14.36.3.38 GetValueAsBool()

```
bool AAX_CStatelessParameter::GetValueAsBool (
    bool * value ) const [inline], [virtual]
```

Retrieves the parameter's value as a bool.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.39 GetValueAsInt32()

```
bool AAX_CStatelessParameter::GetValueAsInt32 (
    int32_t * value ) const [inline], [virtual]
```

Retrieves the parameter's value as an int32_t.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to int32_t was successful
<i>false</i>	The conversion to int32_t was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.40 GetValueAsFloat()

```
bool AAX_CStatelessParameter::GetValueAsFloat (
    float * value ) const [inline], [virtual]
```

Retrieves the parameter's value as a float.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.41 GetValueAsDouble()

```
bool AAX_CStatelessParameter::GetValueAsDouble (
    double * value ) const [inline], [virtual]
```

Retrieves the parameter's value as a double.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to double was successful
<i>false</i>	The conversion to double was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.42 GetValueAsString()

```
bool AAX_CStatelessParameter::GetValueAsString (
    AAX_IString * value ) const [inline], [virtual]
```

Retrieves the parameter's value as a string.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to string was successful
<i>false</i>	The conversion to string was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.43 SetValueWithBool()

```
bool AAX_CStatelessParameter::SetValueWithBool (
    bool value ) [inline], [virtual]
```

Sets the parameter's value as a bool.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion from bool was successful
false	The conversion from bool was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.44 SetValueWithInt32()

```
bool AAX_CStatelessParameter::SetValueWithInt32 (
    int32_t value ) [inline], [virtual]
```

Sets the parameter's value as an int32_t.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion from int32_t was successful
false	The conversion from int32_t was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.45 SetValueWithFloat()

```
bool AAX_CStatelessParameter::SetValueWithFloat (
    float value ) [inline], [virtual]
```

Sets the parameter's value as a float.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion from float was successful
false	The conversion from float was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.46 SetValueWithDouble()

```
bool AAX_CStatelessParameter::SetValueWithDouble (
    double value ) [inline], [virtual]
```

Sets the parameter's value as a double.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from double was successful
<i>false</i>	The conversion from double was unsuccessful

Implements [AAX_IParameter](#).

14.36.3.47 SetValueWithString()

```
bool AAX_CStatelessParameter::SetValueWithString (
    const AAX\_IString & value ) [inline], [virtual]
```

Sets the parameter's value as a string.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from string was successful
<i>false</i>	The conversion from string was unsuccessful

Implements [AAX_IParameter](#).

References [mValueString](#).

14.36.3.48 SetType()

```
void AAX_CStatelessParameter::SetType (
    AAX_EParameterType iControlType ) [inline], [virtual]
```

Sets the type of this parameter.

See [GetType](#) for use cases

Parameters

in	<i>iControlType</i>	The parameter's new type as an AAX_EParameterType
----	---------------------	---

Implements [AAX_IParameter](#).

14.36.3.49 GetType()

```
AAX_EParameterType AAX_CStatelessParameter::GetType ( ) const [inline], [virtual]
```

Returns the type of this parameter as an [AAX_EParameterType](#).

Todo Document use cases for control type

Implements [AAX_IParameter](#).

References [AAX_eParameterType_Discrete](#).

14.36.3.50 SetOrientation()

```
void AAX_CStatelessParameter::SetOrientation (
    AAX_EParameterOrientation iOrientation ) [inline], [virtual]
```

Sets the orientation of this parameter.

Parameters

in	<i>iOrientation</i>	The parameter's new orientation
----	---------------------	---------------------------------

Implements [AAX_IParameter](#).

14.36.3.51 GetOrientation()

```
AAX_EParameterOrientation AAX_CStatelessParameter::GetOrientation ( ) const [inline], [virtual]
```

Returns the orientation of this parameter.

Implements [AAX_IParameter](#).

References [AAX_eParameterOrientation_Default](#).

14.36.3.52 SetTaperDelegate()

```
void AAX_CStatelessParameter::SetTaperDelegate (
    AAX_ITaperDelegateBase & inTaperDelegate,
    bool inPreserveValue ) [inline], [virtual]
```

Sets the parameter's taper delegate.

Parameters

in	<i>inTaperDelegate</i>	A reference to the parameter's new taper delegate
in	<i>inPreserveValue</i>	

Todo Document this parameter

Implements [AAX_IParameter](#).

14.36.3.53 SetDisplayDelegate()

```
void AAX_CStatelessParameter::SetDisplayDelegate (
    AAX_IDisplayDelegateBase & inDisplayDelegate ) [inline], [virtual]
```

Sets the parameter's display delegate.

Parameters

in	<i>inDisplayDelegate</i>	A reference to the parameter's new display delegate
----	--------------------------	---

Implements [AAX_IParameter](#).

14.36.3.54 UpdateNormalizedValue()

```
void AAX_CStatelessParameter::UpdateNormalizedValue (
    double newNormalizedValue ) [inline], [virtual]
```

Sets the parameter's state given a normalized value.

This is the second half of the parameter setting operation that is initiated with a call to `SetValue()`. Parameters should not be set directly using this method; instead, use `SetValue()`.

Parameters

in	<i>newNormalizedValue</i>	Normalized value that will be used to set the parameter's new state
----	---------------------------	---

Implements [AAX_IParameter](#).

14.36.4 Member Data Documentation

14.36.4.1 mName

[AAX_CStringAbbreviations](#) AAX_CStatelessParameter::mName [protected]

Referenced by [AddShortenedName\(\)](#), [ClearShortenedNames\(\)](#), [Name\(\)](#), [SetName\(\)](#), and [ShortenedName\(\)](#).

14.36.4.2 mID

[AAX_CString](#) AAX_CStatelessParameter::mID [protected]

Referenced by [Identifier\(\)](#).

14.36.4.3 mAutomationDelegate

[AAX_IAutomationDelegate*](#) AAX_CStatelessParameter::mAutomationDelegate [protected]

Referenced by [Release\(\)](#), [SetAutomationDelegate\(\)](#), [SetName\(\)](#), and [Touch\(\)](#).

14.36.4.4 mValueString

[AAX_CString](#) AAX_CStatelessParameter::mValueString [protected]

Referenced by [GetStringFromNormalizedValue\(\)](#), [GetValueString\(\)](#), [SetValueFromString\(\)](#), and [SetValueWithString\(\)](#).

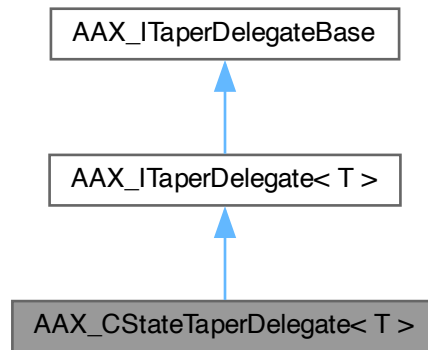
The documentation for this class was generated from the following file:

- [AAX_CParameter.h](#)

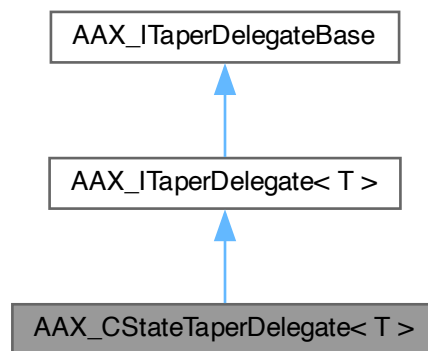
14.37 AAX_CStateTaperDelegate< T > Class Template Reference

```
#include <AAX_CStateTaperDelegate.h>
```

Inheritance diagram for AAX_CStateTaperDelegate< T >:



Collaboration diagram for AAX_CStateTaperDelegate< T >:



14.37.1 Description

```
template<typename T>
class AAX_CStateTaperDelegate< T >
```

A linear taper conforming to [AAX_ITaperDelegate](#).

This taper spaces a parameter's real values evenly between its minimum and maximum, with a linear mapping between the parameter's real and normalized values. It is essentially a version of [AAX_CLinearTaperDelegate](#) without that class' additional RealPrecision templatzation.

Public Member Functions

- [AAX_CStateTaperDelegate](#) (T minValue=0, T maxValue=1)
Constructs a State Taper with specified minimum and maximum values.
- [AAX_CStateTaperDelegate< T > * Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the taper delegate.
- T [GetMinimumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's minimum real value.
- T [GetMaximumValue](#) () const [AAX_OVERRIDE](#)
Returns the taper's maximum real value.
- T [ConstrainRealValue](#) (T value) const [AAX_OVERRIDE](#)
Applies a constraint to the value and returns the constrained value.
- T [NormalizedToReal](#) (double normalizedValue) const [AAX_OVERRIDE](#)
Converts a normalized value to a real value.
- double [RealToNormalized](#) (T realValue) const [AAX_OVERRIDE](#)
Normalizes a real parameter value.

- virtual [AAX_ITaperDelegate * Clone](#) () const =0
Constructs and returns a copy of the taper delegate.
- virtual T [GetMaximumValue](#) () const =0
Returns the taper's maximum real value.
- virtual T [GetMinimumValue](#) () const =0
Returns the taper's minimum real value.
- virtual T [ConstrainRealValue](#) (T value) const =0
Applies a constraint to the value and returns the constrained value.
- virtual T [NormalizedToReal](#) (double normalizedValue) const =0
Converts a normalized value to a real value.
- virtual double [RealToNormalized](#) (T realValue) const =0
Normalizes a real parameter value.

Public Member Functions inherited from [AAX_ITaperDelegateBase](#)

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

14.37.2 Constructor & Destructor Documentation

14.37.2.1 AAX_CStateTaperDelegate()

```
template<typename T >
AAX_CStateTaperDelegate< T >::AAX_CStateTaperDelegate (
    T minValue = 0,
    T maxValue = 1 )
```

Constructs a State Taper with specified minimum and maximum values.

Note

The parameter's default value should lie within the min to max range.

Parameters

in	<i>minValue</i>	
in	<i>maxValue</i>	

14.37.3 Member Function Documentation

14.37.3.1 Clone()

```
template<typename T >
AAX_CStateTaperDelegate< T > * AAX_CStateTaperDelegate< T >::Clone ( ) const [virtual]
```

Constructs and returns a copy of the taper delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassTaperDelegate<T>* AAX_CSubclassTaperDelegate<T>::Clone() const
{
    return new AAX_CSubclassTaperDelegate(*this);
}
```

Implements [AAX_ITaperDelegate< T >](#).

14.37.3.2 GetMinimumValue()

```
template<typename T >
T AAX_CStateTaperDelegate< T >::GetMinimumValue ( ) const [inline], [virtual]
```

Returns the taper's minimum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.37.3.3 GetMaximumValue()

```
template<typename T >
T AAX_CStateTaperDelegate< T >::GetMaximumValue ( ) const [inline], [virtual]
```

Returns the taper's maximum real value.

Implements [AAX_ITaperDelegate< T >](#).

14.37.3.4 ConstrainRealValue()

```
template<typename T >
T AAX_CStateTaperDelegate< T >::ConstrainRealValue (
    T value ) const [virtual]
```

Applies a constraint to the value and returns the constrained value.

This method is useful if the taper requires a constraint beyond simple minimum and maximum real value limits.

Note

This is the function that should actually enforces the constraints in [NormalizeToReal\(\)](#) and [RealToNormalized\(\)](#).

Parameters

in	<i>value</i>	The unconstrained value
----	--------------	-------------------------

Implements [AAX_ITaperDelegate< T >](#).

14.37.3.5 NormalizedToReal()

```
template<typename T >
T AAX_CStateTaperDelegate< T >::NormalizedToReal (
    double normalizedValue ) const [virtual]
```

Converts a normalized value to a real value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [RealToNormalized\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>normalizedValue</i>	The normalized value that will be converted
----	------------------------	---

Implements [AAX_ITaperDelegate< T >](#).

14.37.3.6 RealToNormalized()

```
template<typename T >
double AAX_CStateTaperDelegate< T >::RealToNormalized (
    T realValue ) const [virtual]
```

Normalizes a real parameter value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [NormalizedToReal\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>realValue</i>	The real parameter value that will be normalized
----	------------------	--

Implements [AAX_ITaperDelegate< T >](#).

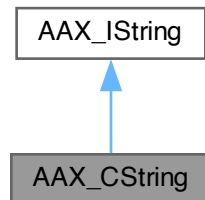
The documentation for this class was generated from the following file:

- [AAX_CStateTaperDelegate.h](#)

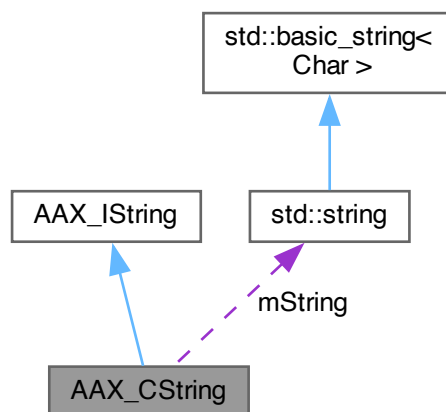
14.38 AAX_CString Class Reference

```
#include <AAX_CString.h>
```

Inheritance diagram for AAX_CString:



Collaboration diagram for AAX_CString:



14.38.1 Description

A generic AAX string class with similar functionality to `std::string`

Public Member Functions

- `uint32_t Length () const` [AAX_OVERRIDE](#)
- `uint32_t MaxLength () const` [AAX_OVERRIDE](#)
- `const char * Get () const` [AAX_OVERRIDE](#)

- void [Set](#) (const char *iString) [AAX_OVERRIDE](#)
- [AAX_IString](#) & [operator=](#) (const [AAX_IString](#) &iOther) [AAX_OVERRIDE](#)
- [AAX_IString](#) & [operator=](#) (const char *iString) [AAX_OVERRIDE](#)
- [AAX_CString](#) ()
- [AAX_CString](#) (const char *str)
- [AAX_CString](#) (const std::string &str)
- [AAX_CString](#) (const [AAX_CString](#) &other)
- [AAX_CString](#) (const [AAX_IString](#) &other)
- [AAX_DEFAULT_MOVE_CTOR](#) ([AAX_CString](#))
- std::string & [StdString](#) ()
- const std::string & [StdString](#) () const
- [AAX_CString](#) & [operator=](#) (const [AAX_CString](#) &other)
- [AAX_CString](#) & [operator=](#) (const std::string &other)
- [AAX_CString](#) & [operator=](#) ([AAX_CString](#) &&other)
- void [Clear](#) ()
- bool [Empty](#) () const
- [AAX_CString](#) & [Erase](#) (uint32_t pos, uint32_t n)
- [AAX_CString](#) & [Append](#) (const [AAX_CString](#) &str)
- [AAX_CString](#) & [Append](#) (const char *str)
- [AAX_CString](#) & [AppendNumber](#) (double number, int32_t precision)
- [AAX_CString](#) & [AppendNumber](#) (int32_t number)
- [AAX_CString](#) & [AppendHex](#) (int32_t number, int32_t width)
- [AAX_CString](#) & [Insert](#) (uint32_t pos, const [AAX_CString](#) &str)
- [AAX_CString](#) & [Insert](#) (uint32_t pos, const char *str)
- [AAX_CString](#) & [InsertNumber](#) (uint32_t pos, double number, int32_t precision)
- [AAX_CString](#) & [InsertNumber](#) (uint32_t pos, int32_t number)
- [AAX_CString](#) & [InsertHex](#) (uint32_t pos, int32_t number, int32_t width)
- [AAX_CString](#) & [Replace](#) (uint32_t pos, uint32_t n, const [AAX_CString](#) &str)
- [AAX_CString](#) & [Replace](#) (uint32_t pos, uint32_t n, const char *str)
- uint32_t [FindFirst](#) (const [AAX_CString](#) &findStr) const
- uint32_t [FindFirst](#) (const char *findStr) const
- uint32_t [FindFirst](#) (char findChar) const
- uint32_t [FindLast](#) (const [AAX_CString](#) &findStr) const
- uint32_t [FindLast](#) (const char *findStr) const
- uint32_t [FindLast](#) (char findChar) const
- const char * [CString](#) () const
- bool [ToDouble](#) (double *oValue) const
- bool [ToInteger](#) (int32_t *oValue) const
- void [SubString](#) (uint32_t pos, uint32_t n, [AAX_IString](#) *outputStr) const
- bool [Equals](#) (const [AAX_CString](#) &other) const
- bool [Equals](#) (const char *other) const
- bool [Equals](#) (const std::string &other) const
- bool [operator==](#) (const [AAX_CString](#) &other) const
- bool [operator==](#) (const char *otherStr) const
- bool [operator==](#) (const std::string &otherStr) const
- bool [operator!=](#) (const [AAX_CString](#) &other) const
- bool [operator!=](#) (const char *otherStr) const
- bool [operator!=](#) (const std::string &otherStr) const
- bool [operator<](#) (const [AAX_CString](#) &other) const
- bool [operator>](#) (const [AAX_CString](#) &other) const
- const char & [operator\[\]](#) (uint32_t index) const
- char & [operator\[\]](#) (uint32_t index)
- [AAX_CString](#) & [operator+=](#) (const [AAX_CString](#) &str)
- [AAX_CString](#) & [operator+=](#) (const std::string &str)
- [AAX_CString](#) & [operator+=](#) (const char *str)

Public Member Functions inherited from [AAX_IString](#)

- virtual [~AAX_IString](#) ()
- virtual uint32_t [Length](#) () const =0
- virtual uint32_t [MaxLength](#) () const =0
- virtual const char * [Get](#) () const =0
- virtual void [Set](#) (const char *iString)=0
- virtual [AAX_IString](#) & [operator=](#) (const [AAX_IString](#) &iOther)=0
- virtual [AAX_IString](#) & [operator=](#) (const char *iString)=0

Static Public Attributes

- static const uint32_t [kInvalidIndex](#) = static_cast<uint32_t>(-1)
- static const uint32_t [kMaxStringLength](#) = static_cast<uint32_t>(-2)

Protected Attributes

- std::string [mString](#)

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [AAX_CString](#) &str)
- std::istream & [operator>>](#) (std::istream &os, [AAX_CString](#) &str)

14.38.2 Constructor & Destructor Documentation

14.38.2.1 [AAX_CString\(\)](#) [1/5]

```
AAX_CString::AAX_CString ( )
```

Constructs an empty string.

14.38.2.2 [AAX_CString\(\)](#) [2/5]

```
AAX_CString::AAX_CString (
    const char * str )
```

Implicit conversion constructor: Constructs a string with a const char* pointer to copy.

14.38.2.3 [AAX_CString\(\)](#) [3/5]

```
AAX_CString::AAX_CString (
    const std::string & str ) [explicit]
```

Copy constructor: Constructs a string from a std::string. Beware of STL variations across various binaries.

14.38.2.4 AAX_CString() [4/5]

```
AAX_CString::AAX_CString (
    const AAX\_CString & other )
```

Copy constructor: Constructs a string with another concrete [AAX_CString](#).

14.38.2.5 AAX_CString() [5/5]

```
AAX_CString::AAX_CString (
    const AAX\_IString & other )
```

Copy constructor: Constructs a string from another string that meets the [AAX_IString](#) interface.

14.38.3 Member Function Documentation

14.38.3.1 Length()

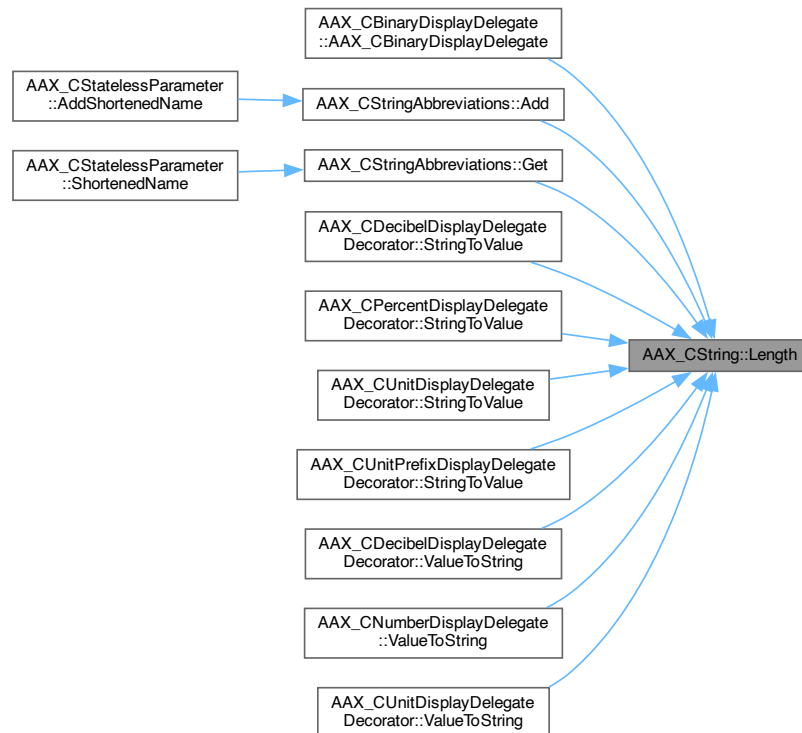
```
uint32_t AAX_CString::Length ( ) const [virtual]
```

Length methods

Implements [AAX_IString](#).

Referenced by [AAX_CBinaryDisplayDelegate< T >::AAX_CBinaryDisplayDelegate\(\)](#), [AAX_CStringAbbreviations::Add\(\)](#), [AAX_CStringAbbreviations::Get\(\)](#), [AAX_CDecibelDisplayDelegateDecorator< T >::StringToValue\(\)](#), [AAX_CPercentDisplayDelegateDecorator< T >::StringToValue\(\)](#), [AAX_CUnitDisplayDelegateDecorator< T >::StringToValue\(\)](#), [AAX_CUnitPrefixDisplayDelegateDecorator< T >::StringToValue\(\)](#), [AAX_CDecibelDisplayDelegateDecorator< T >::ValueToString\(\)](#), [AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >::ValueToString\(\)](#), and [AAX_CUnitDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the caller graph for this function:



14.38.3.2 MaxLength()

```
uint32_t AAX_CString::MaxLength ( ) const [virtual]
```

Implements [AAX_IString](#).

14.38.3.3 Get()

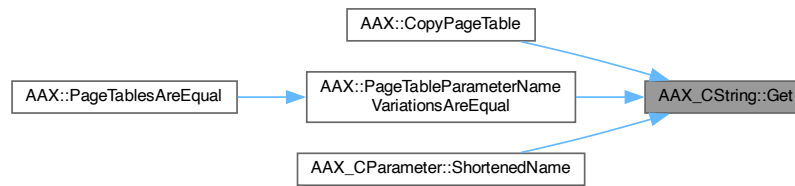
```
const char * AAX_CString::Get ( ) const [virtual]
```

C string methods

Implements [AAX_IString](#).

Referenced by [AAX::CopyPageTable\(\)](#), [AAX::PageTableParameterNameVariationsAreEqual\(\)](#), and [AAX_CParameter< T >::ShortenedName](#).

Here is the caller graph for this function:



14.38.3.4 Set()

```
void AAX_CString::Set (
    const char * iString ) [virtual]
```

Implements [AAX_IString](#).

14.38.3.5 operator=() [1/5]

```
AAX\_IString & AAX_CString::operator= (
    const AAX\_IString & iOther ) [virtual]
```

Assignment operators

Implements [AAX_IString](#).

14.38.3.6 operator=() [2/5]

```
AAX\_IString & AAX_CString::operator= (
    const char * iString ) [virtual]
```

Implements [AAX_IString](#).

14.38.3.7 AAX_DEFAULT_MOVE_CTOR()

```
AAX_CString::AAX_DEFAULT_MOVE_CTOR (
    AAX\_CString )
```

Default move constructor

14.38.3.8 StdString() [1/2]

```
std::string & AAX_CString::StdString ( )
```

Direct access to a std::string.

14.38.3.9 StdString() [2/2]

```
const std::string & AAX_CString::StdString ( ) const
```

Direct access to a const std::string.

14.38.3.10 operator=() [3/5]

```
AAX_CString & AAX_CString::operator= (
    const AAX_CString & other )
```

Assignment operator from another [AAX_CString](#)

14.38.3.11 operator=() [4/5]

```
AAX_CString & AAX_CString::operator= (
    const std::string & other )
```

Assignment operator from a std::string. Beware of STL variations across various binaries.

14.38.3.12 operator=() [5/5]

```
AAX_CString & AAX_CString::operator= (
    AAX_CString && other )
```

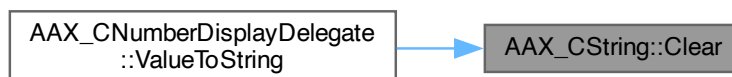
Move operator

14.38.3.13 Clear()

```
void AAX_CString::Clear ( )
```

Referenced by [AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >::ValueToString\(\)](#).

Here is the caller graph for this function:



14.38.3.14 Empty()

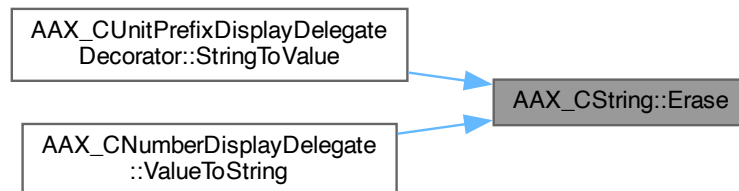
```
bool AAX_CString::Empty ( ) const
```

14.38.3.15 Erase()

```
AAX_CString & AAX_CString::Erase (
    uint32_t pos,
    uint32_t n )
```

Referenced by [AAX_CUnitPrefixDisplayDelegateDecorator< T >::StringToValue\(\)](#), and [AAX_CNumberDisplayDelegate< T, Precision>::ValueToString\(\)](#).

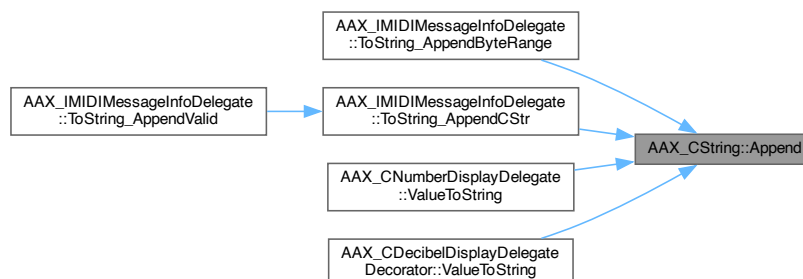
Here is the caller graph for this function:

**14.38.3.16 Append()** [1/2]

```
AAX_CString & AAX_CString::Append (
    const AAX_CString & str )
```

Referenced by [AAX_IMIDIMessageInfoDelegate::ToString_AppendByteRange\(\)](#), [AAX_IMIDIMessageInfoDelegate::ToString_AppendCStr\(\)](#), [AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >::ValueToString\(\)](#), and [AAX_CDecibelDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the caller graph for this function:



14.38.3.17 Append() [2/2]

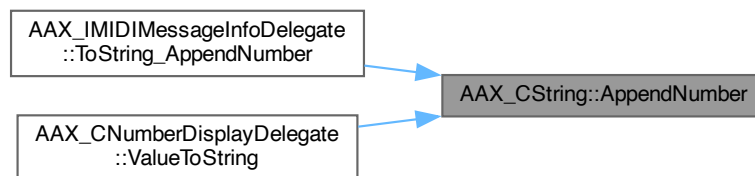
```
AAX_CString & AAX_CString::Append (
    const char * str )
```

14.38.3.18 AppendNumber() [1/2]

```
AAX_CString & AAX_CString::AppendNumber (
    double number,
    int32_t precision )
```

Referenced by [AAX_IMIDIMessageInfoDelegate::ToString_AppendNumber\(\)](#), and [AAX_CNumberDisplayDelegate< T, Precision, Spacing>::ToString_AppendNumber\(\)](#).

Here is the caller graph for this function:

**14.38.3.19 AppendNumber()** [2/2]

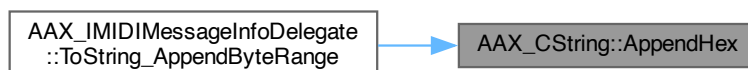
```
AAX_CString & AAX_CString::AppendNumber (
    int32_t number )
```

14.38.3.20 AppendHex()

```
AAX_CString & AAX_CString::AppendHex (
    int32_t number,
    int32_t width )
```

Referenced by [AAX_IMIDIMessageInfoDelegate::ToString_AppendByteRange\(\)](#).

Here is the caller graph for this function:



14.38.3.21 Insert() [1/2]

```
AAX_CString & AAX_CString::Insert (
    uint32_t pos,
    const AAX_CString & str )
```

14.38.3.22 Insert() [2/2]

```
AAX_CString & AAX_CString::Insert (
    uint32_t pos,
    const char * str )
```

14.38.3.23 InsertNumber() [1/2]

```
AAX_CString & AAX_CString::InsertNumber (
    uint32_t pos,
    double number,
    int32_t precision )
```

14.38.3.24 InsertNumber() [2/2]

```
AAX_CString & AAX_CString::InsertNumber (
    uint32_t pos,
    int32_t number )
```

14.38.3.25 InsertHex()

```
AAX_CString & AAX_CString::InsertHex (
    uint32_t pos,
    int32_t number,
    int32_t width )
```

14.38.3.26 Replace() [1/2]

```
AAX_CString & AAX_CString::Replace (
    uint32_t pos,
    uint32_t n,
    const AAX_CString & str )
```

14.38.3.27 Replace() [2/2]

```
AAX_CString & AAX_CString::Replace (
    uint32_t pos,
    uint32_t n,
    const char * str )
```

14.38.3.28 FindFirst() [1/3]

```
uint32_t AAX_CString::FindFirst (
    const AAX_CString & findStr ) const
```

14.38.3.29 FindFirst() [2/3]

```
uint32_t AAX_CString::FindFirst (
    const char * findStr ) const
```

14.38.3.30 FindFirst() [3/3]

```
uint32_t AAX_CString::FindFirst (
    char findChar ) const
```

14.38.3.31 FindLast() [1/3]

```
uint32_t AAX_CString::FindLast (
    const AAX_CString & findStr ) const
```

14.38.3.32 FindLast() [2/3]

```
uint32_t AAX_CString::FindLast (
    const char * findStr ) const
```

14.38.3.33 FindLast() [3/3]

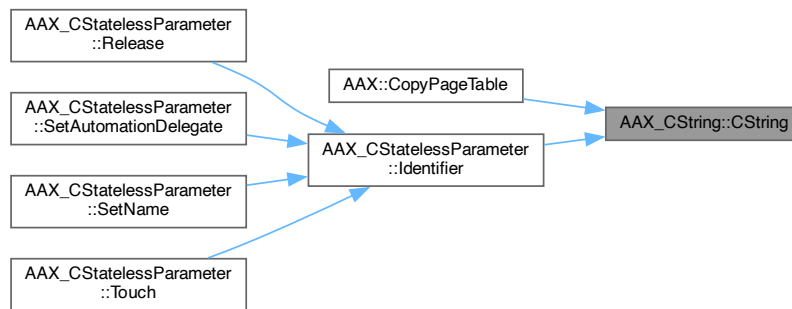
```
uint32_t AAX_CString::FindLast (
    char findChar ) const
```

14.38.3.34 CString()

```
const char * AAX_CString::CString ( ) const
```

Referenced by [AAX::CopyPageTable\(\)](#), and [AAX_CStatelessParameter::Identifier\(\)](#).

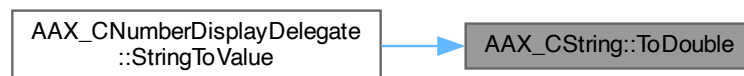
Here is the caller graph for this function:

**14.38.3.35 ToDouble()**

```
bool AAX_CString::ToDouble (
    double * oValue ) const
```

Referenced by [AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >::StringToValue\(\)](#).

Here is the caller graph for this function:

**14.38.3.36 ToInteger()**

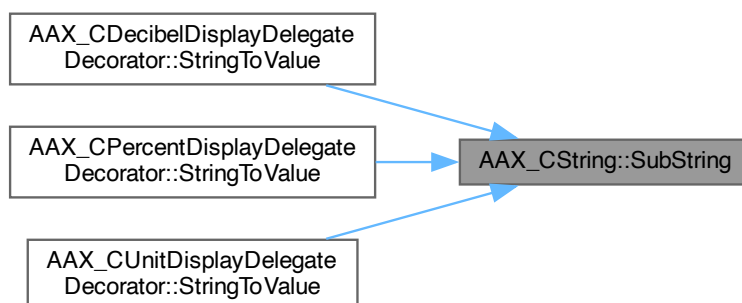
```
bool AAX_CString::ToInteger (
    int32_t * oValue ) const
```

14.38.3.37 SubString()

```
void AAX_CString::SubString (
    uint32_t pos,
    uint32_t n,
    AAX_IString * outputStr ) const
```

Referenced by [AAX_CDecibelDisplayDelegateDecorator< T >::StringToValue\(\)](#), [AAX_CPercentDisplayDelegateDecorator< T >::StringToValue\(\)](#) and [AAX_CUnitDisplayDelegateDecorator< T >::StringToValue\(\)](#).

Here is the caller graph for this function:



14.38.3.38 Equals() [1/3]

```
bool AAX_CString::Equals (
    const AAX_CString & other ) const [inline]
```

References [operator==\(\)](#).

Here is the call graph for this function:

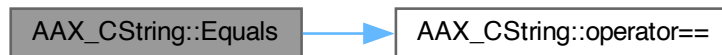


14.38.3.39 Equals() [2/3]

```
bool AAX_CString::Equals (
    const char * other ) const [inline]
```

References [operator==\(\)](#).

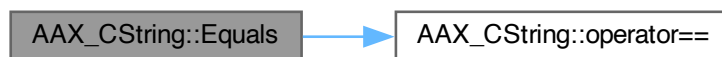
Here is the call graph for this function:

**14.38.3.40 Equals()** [3/3]

```
bool AAX_CString::Equals (
    const std::string & other ) const [inline]
```

References [operator==\(\)](#).

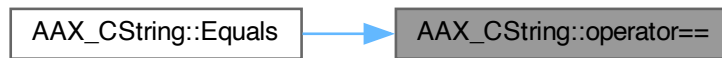
Here is the call graph for this function:

**14.38.3.41 operator==()** [1/3]

```
bool AAX_CString::operator== (
    const AAX\_CString & other ) const
```

Referenced by [Equals\(\)](#).

Here is the caller graph for this function:



14.38.3.42 `operator==()` [2/3]

```
bool AAX_CString::operator==(
    const char * otherStr ) const
```

14.38.3.43 `operator==()` [3/3]

```
bool AAX_CString::operator==(
    const std::string & otherStr ) const
```

14.38.3.44 `operator!=()` [1/3]

```
bool AAX_CString::operator!=(
    const AAX\_CString & other ) const
```

14.38.3.45 `operator!=()` [2/3]

```
bool AAX_CString::operator!=(
    const char * otherStr ) const
```

14.38.3.46 `operator!=()` [3/3]

```
bool AAX_CString::operator!=(
    const std::string & otherStr ) const
```

14.38.3.47 operator<()

```
bool AAX_CString::operator< (
    const AAX_CString & other ) const
```

14.38.3.48 operator>()

```
bool AAX_CString::operator> (
    const AAX_CString & other ) const
```

14.38.3.49 operator[]() [1/2]

```
const char & AAX_CString::operator[] (
    uint32_t index ) const
```

14.38.3.50 operator[]() [2/2]

```
char & AAX_CString::operator[] (
    uint32_t index )
```

14.38.3.51 operator+=() [1/3]

```
AAX_CString & AAX_CString::operator+= (
    const AAX_CString & str )
```

14.38.3.52 operator+=() [2/3]

```
AAX_CString & AAX_CString::operator+= (
    const std::string & str )
```

14.38.3.53 operator+=() [3/3]

```
AAX_CString & AAX_CString::operator+= (
    const char * str )
```

14.38.4 Friends And Related Function Documentation

14.38.4.1 operator<<

```
std::ostream & operator<< (  
    std::ostream & os,  
    const AAX_CString & str ) [friend]
```

output stream operator for concrete [AAX_CString](#)

14.38.4.2 operator>>

```
std::istream & operator>> (  
    std::istream & os,  
    AAX_CString & str ) [friend]
```

input stream operator for concrete [AAX_CString](#)

14.38.5 Member Data Documentation

14.38.5.1 kInvalidIndex

```
const uint32_t AAX_CString::kInvalidIndex = static_cast<uint32_t>(-1) [static]
```

14.38.5.2 kMaxStringLength

```
const uint32_t AAX_CString::kMaxStringLength = static_cast<uint32_t>(-2) [static]
```

14.38.5.3 mString

```
std::string AAX_CString::mString [protected]
```

The documentation for this class was generated from the following file:

- [AAX_CString.h](#)

14.39 AAX_CStringAbbreviations Class Reference

```
#include <AAX_CString.h>
```

14.39.1 Description

Helper class to store a collection of name abbreviations.

Public Member Functions

- [AAX_CStringAbbreviations](#) (const [AAX_CString](#) &inPrimary)
- void [SetPrimary](#) (const [AAX_CString](#) &inPrimary)
- const [AAX_CString](#) & [Primary](#) () const
- void [Add](#) (const [AAX_CString](#) &inAbbreviation)
- const [AAX_CString](#) & [Get](#) (int32_t inNumCharacters) const
- void [Clear](#) ()

14.39.2 Constructor & Destructor Documentation

14.39.2.1 AAX_CStringAbbreviations()

```
AAX_CStringAbbreviations::AAX_CStringAbbreviations (
    const AAX\_CString & inPrimary ) [inline], [explicit]
```

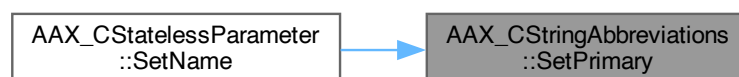
14.39.3 Member Function Documentation

14.39.3.1 SetPrimary()

```
void AAX_CStringAbbreviations::SetPrimary (
    const AAX\_CString & inPrimary ) [inline]
```

Referenced by [AAX_CStatelessParameter::SetName\(\)](#).

Here is the caller graph for this function:

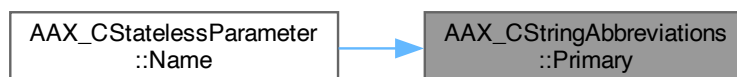


14.39.3.2 Primary()

```
const AAX_CString & AAX_CStringAbbreviations::Primary ( ) const [inline]
```

Referenced by [AAX_CStatelessParameter::Name\(\)](#).

Here is the caller graph for this function:



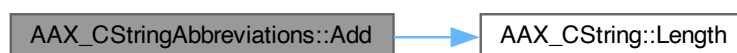
14.39.3.3 Add()

```
void AAX_CStringAbbreviations::Add (
    const AAX_CString & inAbbreviation ) [inline]
```

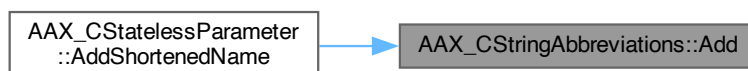
References [AAX_CString::Length\(\)](#).

Referenced by [AAX_CStatelessParameter::AddShortenedName\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.39.3.4 Get()

```
const AAX_CString & AAX_CStringAbbreviations::Get (
    int32_t inNumCharacters ) const [inline]
```

References [AAX_CString::Length\(\)](#).

Referenced by [AAX_CStatelessParameter::ShortenedName\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

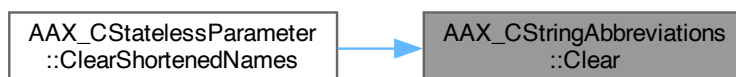


14.39.3.5 Clear()

```
void AAX_CStringAbbreviations::Clear ( ) [inline]
```

Referenced by [AAX_CStatelessParameter::ClearShortenedNames\(\)](#).

Here is the caller graph for this function:



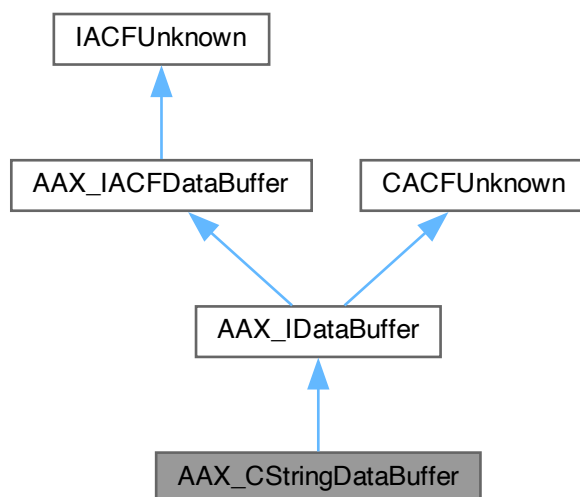
The documentation for this class was generated from the following file:

- [AAX_CString.h](#)

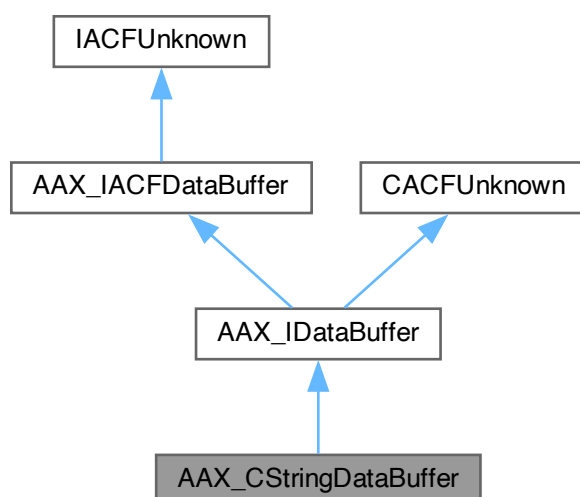
14.40 AAX_CStringDataBuffer Class Reference

```
#include <AAX_CStringDataBuffer.h>
```

Inheritance diagram for AAX_CStringDataBuffer:



Collaboration diagram for AAX_CStringDataBuffer:



14.40.1 Description

A convenience class for string data buffers.

The data payload is a `char*` C string

Public Member Functions

- [AAX_CStringDataBuffer](#) ([AAX_CTypeID](#) inType, std::string const &inData)
- [AAX_CStringDataBuffer](#) ([AAX_CTypeID](#) inType, std::string &&inData)
- [AAX_CStringDataBuffer](#) ([AAX_CTypeID](#) inType, const char *inData)
- [AAX_CStringDataBuffer](#) ([AAX_CStringDataBuffer](#) const &)=delete
- [AAX_CStringDataBuffer](#) ([AAX_CStringDataBuffer](#) &&)=delete
- [~AAX_CStringDataBuffer](#) (void) [AAX_OVERRIDE](#)=default
- [AAX_CStringDataBuffer](#) & operator= ([AAX_CStringDataBuffer](#) const &other)=delete
- [AAX_CStringDataBuffer](#) & operator= ([AAX_CStringDataBuffer](#) &&other)=delete
- [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const [AAX_OVERRIDE](#)
- [AAX_Result Size](#) (int32_t *oSize) const [AAX_OVERRIDE](#)
- [AAX_Result Data](#) (void const **oBuffer) const [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_IDataBuffer](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfIID](#) &riid)
- [AAX_DELETE](#) ([AAX_IDataBuffer](#) &operator=(const [AAX_IDataBuffer](#) &))
- virtual [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result Size](#) (int32_t *oSize) const =0
- virtual [AAX_Result Data](#) (void const **oBuffer) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Additional Inherited Members

Public Attributes inherited from [AAX_IDataBuffer](#)

- void **ppvObjOut [AAX_OVERRIDE](#)

14.40.2 Constructor & Destructor Documentation

14.40.2.1 AAX_CStringDataBuffer() [1/5]

```
AAX_CStringDataBuffer::AAX_CStringDataBuffer (
    AAX_CTypeID inType,
    std::string const & inData ) [inline]
```

14.40.2.2 AAX_CStringDataBuffer() [2/5]

```
AAX_CStringDataBuffer::AAX_CStringDataBuffer (
    AAX_CTypeID inType,
    std::string && inData ) [inline]
```

14.40.2.3 AAX_CStringDataBuffer() [3/5]

```
AAX_CStringDataBuffer::AAX_CStringDataBuffer (
    AAX_CTypeID inType,
    const char * inData ) [inline]
```

14.40.2.4 AAX_CStringDataBuffer() [4/5]

```
AAX_CStringDataBuffer::AAX_CStringDataBuffer (
    AAX_CStringDataBuffer const & ) [delete]
```

14.40.2.5 AAX_CStringDataBuffer() [5/5]

```
AAX_CStringDataBuffer::AAX_CStringDataBuffer (
    AAX_CStringDataBuffer && ) [delete]
```

14.40.2.6 ~AAX_CStringDataBuffer()

```
AAX_CStringDataBuffer::~AAX_CStringDataBuffer (
    void ) [default]
```

14.40.3 Member Function Documentation

14.40.3.1 operator=() [1/2]

```
AAX_CStringDataBuffer & AAX_CStringDataBuffer::operator= (
    AAX_CStringDataBuffer const & other ) [delete]
```

14.40.3.2 operator=() [2/2]

```
AAX_CStringDataBuffer & AAX_CStringDataBuffer::operator= (
    AAX_CStringDataBuffer && other ) [delete]
```

14.40.3.3 Type()

```
AAX_Result AAX_CStringDataBuffer::Type (
    AAX_CTypeID * oType ) const [inline], [virtual]
```

The type of data contained in this buffer

This identifier must be sufficient for a client that knows the type to correctly interpret and use the data.

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

14.40.3.4 Size()

```
AAX_Result AAX_CStringDataBuffer::Size (
    int32_t * oSize ) const [inline], [virtual]
```

The number of bytes of data in this buffer

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), [AAX_ERROR_SIGNED_INT_OVERFLOW](#), and [AAX_SUCCESS](#).

14.40.3.5 Data()

```
AAX_Result AAX_CStringDataBuffer::Data (
    void const ** oBuffer ) const [inline], [virtual]
```

The buffer of data

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

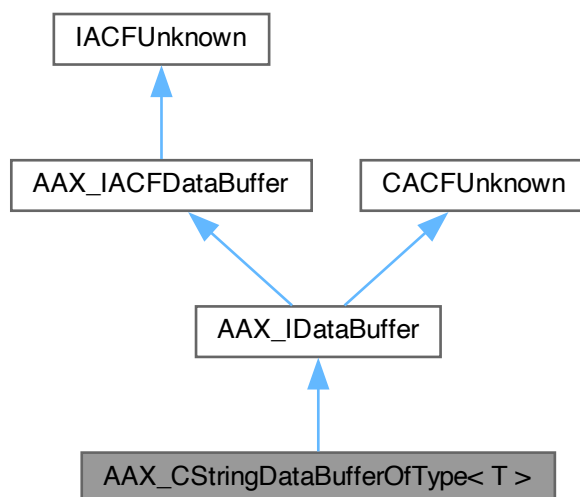
The documentation for this class was generated from the following file:

- [AAX_CStringDataBuffer.h](#)

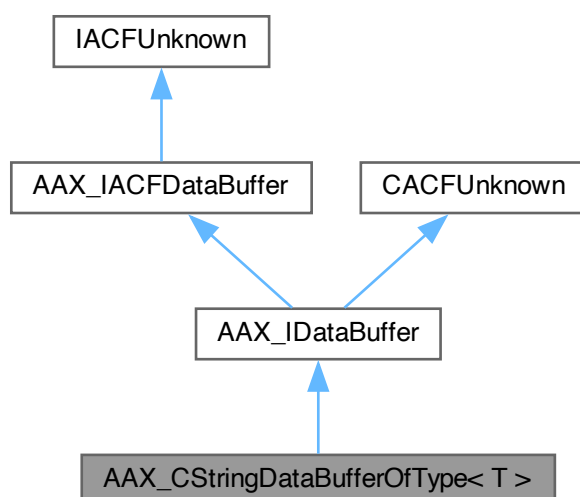
14.41 AAX_CStringDataBufferOfType< T > Class Template Reference

```
#include <AAX_CStringDataBuffer.h>
```

Inheritance diagram for AAX_CStringDataBufferOfType< T >:



Collaboration diagram for AAX_CStringDataBufferOfType< T >:



14.41.1 Description

```
template<AAX\_CTypeID T>
class AAX_CStringDataBufferOfType< T >
```

A convenience class for string data buffers.

The data payload is a `char*` C string

Public Member Functions

- [AAX_CStringDataBufferOfType](#) (std::string const &inData)
- [AAX_CStringDataBufferOfType](#) (std::string &&inData)
- [AAX_CStringDataBufferOfType](#) (const char *inData)
- [AAX_CStringDataBufferOfType](#) ([AAX_CStringDataBufferOfType](#) const &)=delete
- [AAX_CStringDataBufferOfType](#) ([AAX_CStringDataBufferOfType](#) &&)=delete
- [~AAX_CStringDataBufferOfType](#) (void) [AAX_OVERRIDE](#)=default
- [AAX_CStringDataBufferOfType](#) & operator= ([AAX_CStringDataBufferOfType](#) const &other)=delete
- [AAX_CStringDataBufferOfType](#) & operator= ([AAX_CStringDataBufferOfType](#) &&other)=delete
- [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const [AAX_OVERRIDE](#)
- [AAX_Result Size](#) (int32_t *oSize) const [AAX_OVERRIDE](#)
- [AAX_Result Data](#) (void const **oBuffer) const [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_IDataBuffer](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid)
- [AAX_DELETE](#) ([AAX_IDataBuffer](#) &operator=(const [AAX_IDataBuffer](#) &))
- virtual [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result Size](#) (int32_t *oSize) const =0
- virtual [AAX_Result Data](#) (void const **oBuffer) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Additional Inherited Members

Public Attributes inherited from [AAX_IDataBuffer](#)

- void **ppvObjOut [AAX_OVERRIDE](#)

14.41.2 Constructor & Destructor Documentation

14.41.2.1 AAX_CStringDataBufferOfType() [1/5]

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType< T >::AAX_CStringDataBufferOfType (
    std::string const & inData ) [inline], [explicit]
```

14.41.2.2 AAX_CStringDataBufferOfType() [2/5]

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType< T >::AAX_CStringDataBufferOfType (
    std::string && inData ) [inline], [explicit]
```

14.41.2.3 AAX_CStringDataBufferOfType() [3/5]

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType< T >::AAX_CStringDataBufferOfType (
    const char * inData ) [inline], [explicit]
```

14.41.2.4 AAX_CStringDataBufferOfType() [4/5]

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType< T >::AAX_CStringDataBufferOfType (
    AAX_CStringDataBufferOfType< T > const & ) [delete]
```

14.41.2.5 AAX_CStringDataBufferOfType() [5/5]

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType< T >::AAX_CStringDataBufferOfType (
    AAX_CStringDataBufferOfType< T > && ) [delete]
```

14.41.2.6 ~AAX_CStringDataBufferOfType()

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType< T >::~~AAX_CStringDataBufferOfType (
    void ) [default]
```

14.41.3 Member Function Documentation

14.41.3.1 operator=() [1/2]

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType & AAX_CStringDataBufferOfType< T >::operator= (
    AAX_CStringDataBufferOfType< T > const & other ) [delete]
```

14.41.3.2 operator=() [2/2]

```
template<AAX_CTypeID T>
AAX_CStringDataBufferOfType & AAX_CStringDataBufferOfType< T >::operator= (
    AAX_CStringDataBufferOfType< T > && other ) [delete]
```

14.41.3.3 Type()

```
template<AAX_CTypeID T>
AAX_Result AAX_CStringDataBufferOfType< T >::Type (
    AAX_CTypeID * oType ) const [inline], [virtual]
```

The type of data contained in this buffer

This identifier must be sufficient for a client that knows the type to correctly interpret and use the data.

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

14.41.3.4 Size()

```
template<AAX_CTypeID T>
AAX_Result AAX_CStringDataBufferOfType< T >::Size (
    int32_t * oSize ) const [inline], [virtual]
```

The number of bytes of data in this buffer

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), [AAX_ERROR_SIGNED_INT_OVERFLOW](#), and [AAX_SUCCESS](#).

14.41.3.5 Data()

```
template<AAX_CTypeID T>
AAX_Result AAX_CStringDataBufferOfType< T >::Data (
    void const ** oBuffer ) const [inline], [virtual]
```

The buffer of data

Implements [AAX_IACFDataBuffer](#).

References [AAX_ERROR_NULL_ARGUMENT](#), and [AAX_SUCCESS](#).

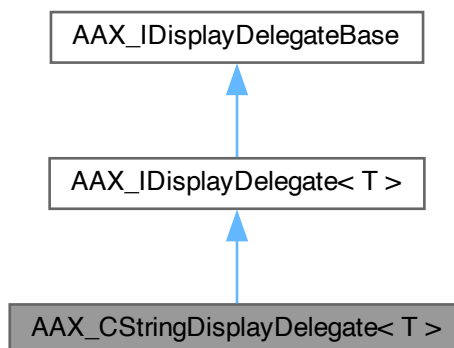
The documentation for this class was generated from the following file:

- [AAX_CStringDataBuffer.h](#)

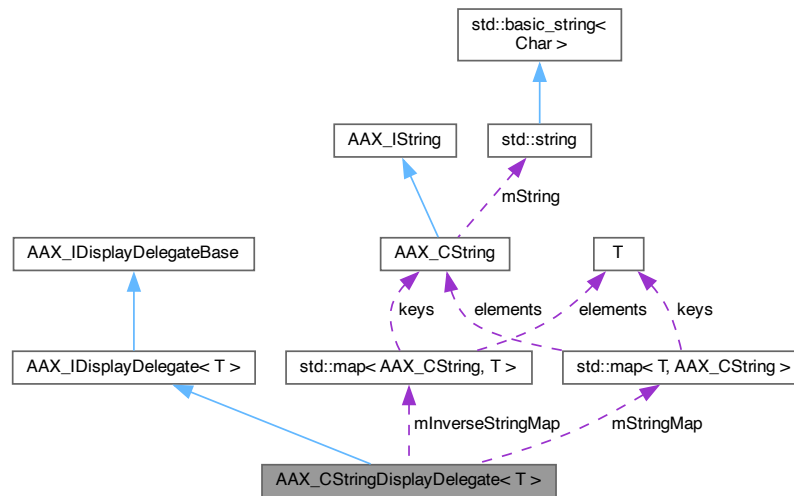
14.42 AAX_CStringDisplayDelegate< T > Class Template Reference

```
#include <AAX_CStringDisplayDelegate.h>
```

Inheritance diagram for AAX_CStringDisplayDelegate< T >:



Collaboration diagram for AAX_CStringDisplayDelegate< T >:



14.42.1 Description

```
template<typename T>
class AAX_CStringDisplayDelegate< T >
```

A string, or list, display format conforming to [AAX_IDisplayDelegate](#).

This display delegate uses a string map to associate parameter values with specific strings. This kind of display delegate is most often used for control string or list parameters, which would internally use an integer parameter type. The int value would then be used as a lookup into this delegate, which would return a string for each valid int value.

Public Member Functions

- [AAX_CStringDisplayDelegate](#) (const std::map< T, [AAX_CString](#) > &stringMap)
Constructor.
- [AAX_CStringDisplayDelegate< T > * Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- virtual [AAX_IDisplayDelegate * Clone](#) () const =0
Constructs and returns a copy of the display delegate.

- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

Protected Attributes

- std::map< T, [AAX_CString](#) > mStringMap
- std::map< [AAX_CString](#), T > mInverseStringMap

14.42.2 Constructor & Destructor Documentation

14.42.2.1 [AAX_CStringDisplayDelegate](#)()

```
template<typename T >
AAX\_CStringDisplayDelegate< T >::AAX_CStringDisplayDelegate (
    const std::map< T, AAX\_CString > & stringMap )
```

Constructor.

Constructs a String Display Delegate with a provided string map.

Note

The string map should already be populated with value-string pairs, as this constructor will copy the provided map into the delegate object's own memory.

Parameters

in	<i>stringMap</i>	A populated map of value-string pairs
----	------------------	---------------------------------------

References [AAX_CStringDisplayDelegate< T >::mInverseStringMap](#), and [AAX_CStringDisplayDelegate< T >::mStringMap](#).

14.42.3 Member Function Documentation

14.42.3.1 Clone()

```
template<typename T >
AAX_CStringDisplayDelegate< T > * AAX_CStringDisplayDelegate< T >::Clone ( ) const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>*    AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.42.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX_CStringDisplayDelegate< T >::ValueToString (
    T value,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.42.3.3 ValueToString() [2/2]

```
template<typename T >
bool AAX_CStringDisplayDelegate< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.42.3.4 StringToValue()

```
template<typename T >
bool AAX_CStringDisplayDelegate< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.42.4 Member Data Documentation**14.42.4.1 mStringMap**

```
template<typename T >
std::map<T, AAX_CString> AAX_CStringDisplayDelegate< T >::mStringMap [protected]
```

Referenced by [AAX_CStringDisplayDelegate< T >::AAX_CStringDisplayDelegate\(\)](#).

14.42.4.2 mInverseStringMap

```
template<typename T >
std::map<AAX_CString, T> AAX_CStringDisplayDelegate< T >::mInverseStringMap [protected]
```

Referenced by [AAX_CStringDisplayDelegate< T >::AAX_CStringDisplayDelegate\(\)](#).

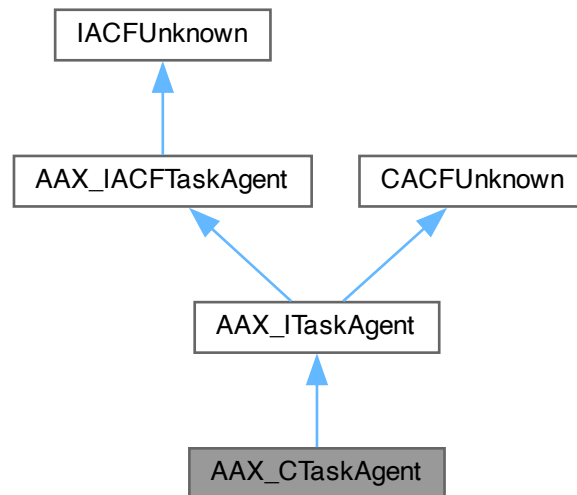
The documentation for this class was generated from the following file:

- [AAX_CStringDisplayDelegate.h](#)

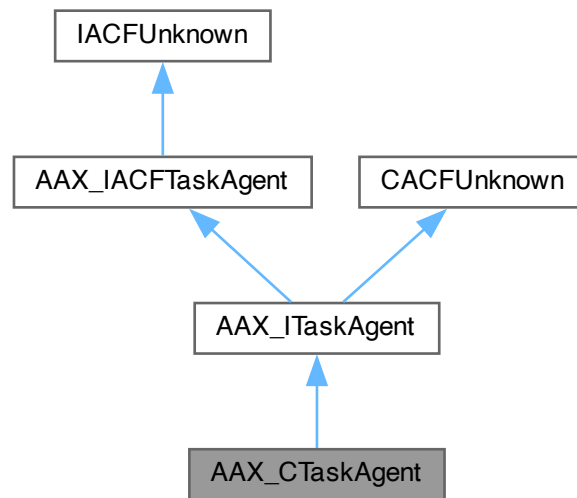
14.43 AAX_CTaskAgent Class Reference

```
#include <AAX_CTaskAgent.h>
```

Inheritance diagram for AAX_CTaskAgent:



Collaboration diagram for AAX_CTaskAgent:



14.43.1 Description

Default implementation of the [AAX_ITaskAgent](#) interface.

This class provides a default implementation of the [AAX_ITaskAgent](#) interface. Your plug-in's task agent implementation should inherit from this class and override the remaining interface functions.

Public Member Functions

- [AAX_CTaskAgent](#) (void)=default
- [~AAX_CTaskAgent](#) (void) [AAX_OVERRIDE](#)

Initialization and uninitialization

- [AAX_Result Initialize](#) ([IACFUnknown](#) *iController) [AAX_OVERRIDE](#)
- [AAX_Result Uninitialize](#) (void) [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_ITaskAgent](#)

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid
- [AAX_DELETE](#) ([AAX_ITaskAgent](#) &operator=(const [AAX_ITaskAgent](#) &))

Initialization and uninitialization

Task management

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Private member accessors

- [AAX_IController](#) * [GetController](#) (void)
Returns a pointer to the plug-in's controller interface.
- [AAX_IEffectParameters](#) * [GetEffectParameters](#) (void)
Returns a pointer to the plug-in's data model interface.

Task management

- [AAX_Result](#) [AddTask](#) ([IACFUnknown](#) *iTask) [AAX_OVERRIDE](#)
Default implementation of [AddTask\(\)](#)
- [AAX_Result](#) [CancelAllTasks](#) () [AAX_OVERRIDE](#)
- virtual [AAX_Result](#) [AddTask](#) (std::unique_ptr< [AAX_ITask](#) > iTask)
Convenience method for adding versioned tasks.
- virtual [AAX_Result](#) [ReceiveTask](#) (std::unique_ptr< [AAX_ITask](#) > iTask)
Convenience method for adding versioned tasks.

Additional Inherited Members**Public Attributes inherited from [AAX_ITaskAgent](#)**

- void **ppvObjOut [AAX_OVERRIDE](#)

14.43.2 Constructor & Destructor Documentation**14.43.2.1 [AAX_CTaskAgent](#)()**

```
AAX_CTaskAgent::AAX_CTaskAgent (
    void ) [default]
```

14.43.2.2 ~AAX_CTaskAgent()

```
AAX_CTaskAgent::~~AAX_CTaskAgent (
    void )
```

14.43.3 Member Function Documentation

14.43.3.1 Initialize()

```
AAX_Result AAX_CTaskAgent::Initialize (
    IACFUnknown * iController ) [virtual]
```

Initialize the object

Parameters

in	<i>iController</i>	Interface allowing access to other objects in the object graph such as the plug-in's data model.
----	--------------------	--

Implements [AAX_IACFTaskAgent](#).

14.43.3.2 Uninitialize()

```
AAX_Result AAX_CTaskAgent::Uninitialize (
    void ) [virtual]
```

Uninitialize the object

This method should release references to any shared objects

Implements [AAX_IACFTaskAgent](#).

14.43.3.3 AddTask() [1/2]

```
AAX_Result AAX_CTaskAgent::AddTask (
    IACFUnknown * iTask ) [virtual]
```

Default implemenation of [AddTask\(\)](#)

Convenience implementation that converts the [IACFUnknown](#) into an [AAX_ITask](#) . Implementations should override the version that provides an [AAX_ITask](#) object.

Implements [AAX_IACFTaskAgent](#).

14.43.3.4 CancelAllTasks()

```
AAX_Result AAX_CTaskAgent::CancelAllTasks ( ) [virtual]
```

Request that the agent cancel all outstanding tasks

Implements [AAX_IACFTaskAgent](#).

14.43.3.5 AddTask() [2/2]

```
virtual AAX_Result AAX_CTaskAgent::AddTask (
    std::unique_ptr< AAX_ITask > iTask ) [protected], [virtual]
```

Convenience method for adding versioned tasks.

Deprecated Use [ReceiveTask\(\)](#) instead

14.43.3.6 ReceiveTask()

```
virtual AAX_Result AAX_CTaskAgent::ReceiveTask (
    std::unique_ptr< AAX_ITask > iTask ) [protected], [virtual]
```

Convenience method for adding versioned tasks.

14.43.3.7 GetController()

```
AAX_IController * AAX_CTaskAgent::GetController (
    void ) [inline]
```

Returns a pointer to the plug-in's controller interface.

14.43.3.8 GetEffectParameters()

```
AAX_IEffectParameters * AAX_CTaskAgent::GetEffectParameters (
    void ) [inline]
```

Returns a pointer to the plug-in's data model interface.

The documentation for this class was generated from the following file:

- [AAX_CTaskAgent.h](#)

14.44 AAX_CTempoBreakpoint Struct Reference

```
#include <AAX_SessionDocumentTypes.h>
```

Public Attributes

- `int64_t mSampleLocation {0}`
- `float mValue {0.f}`

14.44.1 Member Data Documentation

14.44.1.1 mSampleLocation

```
int64_t AAX_CTempoBreakpoint::mSampleLocation {0}
```

14.44.1.2 mValue

```
float AAX_CTempoBreakpoint::mValue {0.f}
```

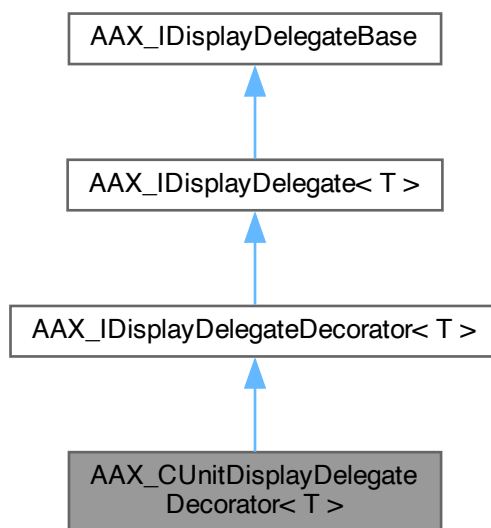
The documentation for this struct was generated from the following file:

- [AAX_SessionDocumentTypes.h](#)

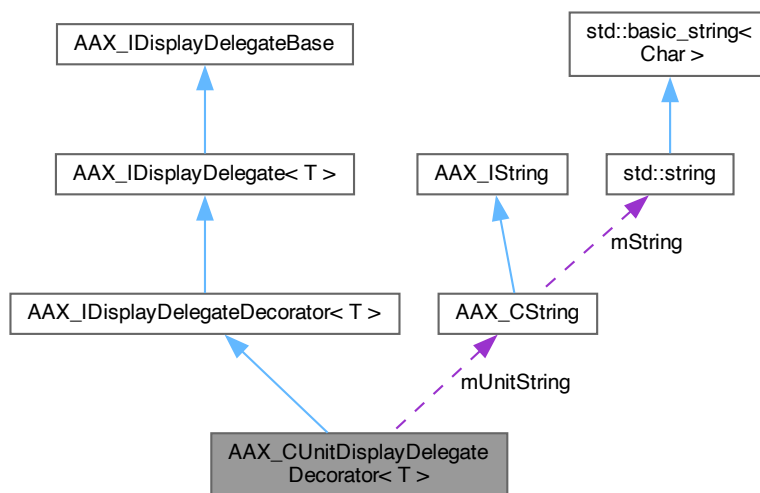
14.45 AAX_CUnitDisplayDelegateDecorator< T > Class Template Reference

```
#include <AAX_CUnitDisplayDelegateDecorator.h>
```

Inheritance diagram for AAX_CUnitDisplayDelegateDecorator< T >:



Collaboration diagram for AAX_CUnitDisplayDelegateDecorator< T >:



14.45.1 Description

```
template<typename T>
class AAX_CUnitDisplayDelegateDecorator< T >
```

A unit type decorator conforming to [AAX_IDisplayDelegateDecorator](#).

This class is an [AAX_IDisplayDelegateDecorator](#), meaning that it acts as a wrapper for other display delegates or concrete display types. For more information about display delegate decorators in [AAX](#), see [Display delegate decorators](#)

The behavior of this class is to decorate parameter value strings with arbitrary units, such as "Hz" or "V". The inverse is also supported, so the unit string is pulled off of value strings when they are converted to real parameter values.

Public Member Functions

- [AAX_CUnitDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate, const [AAX_CString](#) &unitString)
Constructor.
- [AAX_CUnitDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateDecorator](#)< T >

- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate)
Constructor.
- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegateDecorator](#) &other)
Copy constructor.
- [~AAX_IDisplayDelegateDecorator](#) () [AAX_OVERRIDE](#)
Virtual destructor.
- [AAX_IDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate decorator.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value with a size constraint.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

Protected Attributes

- const [AAX_CString](#) mUnitString

14.45.2 Constructor & Destructor Documentation**14.45.2.1 AAX_CUnitDisplayDelegateDecorator()**

```
template<typename T >
AAX_CUnitDisplayDelegateDecorator< T >::AAX_CUnitDisplayDelegateDecorator (
    const AAX\_IDisplayDelegate< T > & displayDelegate,
    const AAX\_CString & unitString )
```

Constructor.

Along with the standard decorator pattern argument, this class also takes a unit string. This is the string that will be added to the end of valueString.

Parameters

in	<i>displayDelegate</i>	
in	<i>unitString</i>	

14.45.3 Member Function Documentation**14.45.3.1 Clone()**

```
template<typename T >
AAX_CUnitDisplayDelegateDecorator< T > * AAX\_CUnitDisplayDelegateDecorator< T >::Clone ( )
const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template<typename T>
AAX_CSubclassDisplayDelegate<T>*    AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.45.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX_CUnitDisplayDelegateDecorator< T >::ValueToString (
    T value,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to value

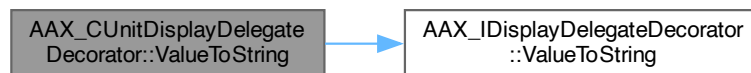
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:

**14.45.3.3 ValueToString()** [2/2]

```
template<typename T >
bool AAX_CUnitDisplayDelegateDecorator< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

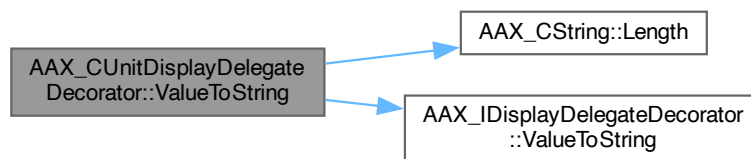
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Length\(\)](#), and [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:



14.45.3.4 StringToValue()

```

template<typename T >
bool AAX_CUnitDisplayDelegateDecorator< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
  
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

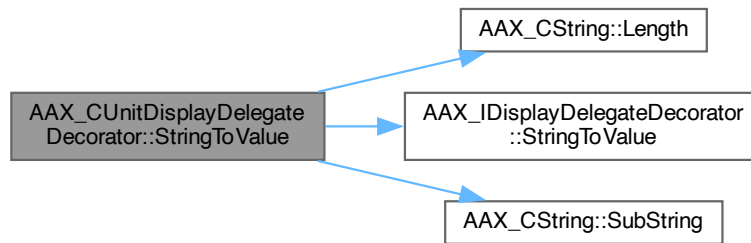
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Length\(\)](#), [AAX_IDisplayDelegateDecorator< T >::StringToValue\(\)](#), and [AAX_CString::SubString\(\)](#).

Here is the call graph for this function:



14.45.4 Member Data Documentation

14.45.4.1 mUnitString

```
template<typename T >
const AAX_CString AAX_CUnitDisplayDelegateDecorator< T >::mUnitString [protected]
```

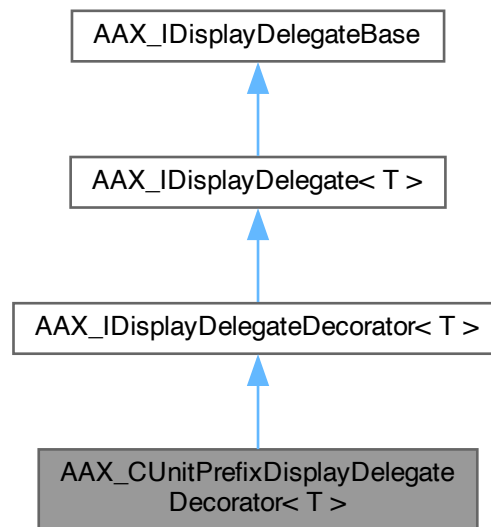
The documentation for this class was generated from the following file:

- [AAX_CUnitDisplayDelegateDecorator.h](#)

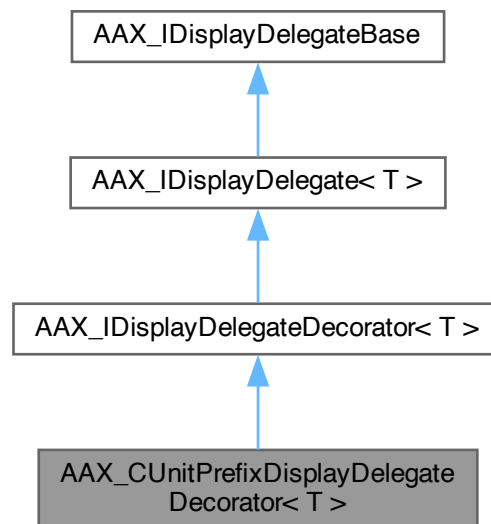
14.46 AAX_CUnitPrefixDisplayDelegateDecorator< T > Class Template Reference

```
#include <AAX_CUnitPrefixDisplayDelegateDecorator.h>
```


Inheritance diagram for AAX_CUnitPrefixDisplayDelegateDecorator< T >:



Collaboration diagram for AAX_CUnitPrefixDisplayDelegateDecorator< T >:



14.46.1 Description

```
template<typename T>
class AAX_CUnitPrefixDisplayDelegateDecorator< T >
```

A unit prefix decorator conforming to [AAX_IDisplayDelegateDecorator](#).

This class is an [AAX_IDisplayDelegateDecorator](#), meaning that it acts as a wrapper for other display delegates or concrete display types. For more information about display delegate decorators in [AAX](#), see [Display delegate decorators](#)

The behavior of this class is to provide unit prefixes such as the k in kHz or the m in mm. It takes the value passed in and determines if the value is large or small enough to benefit from a unit modifier. If so, it adds that unit prefix character to the display string after scaling the number and calling deeper into the decorator pattern to get the concrete [ValueToString\(\)](#) result.

The inverse is also supported, so if you type 1.5k in a text box and this decorator is in place, it should find the k and multiply the value by 1000 before converting it to a real value.

This decorator supports the following unit prefixes:

- M (mega-)
- k (kilo-)
- m (milli-)
- u (micro-)

Note

This class is not implemented for integer values as the conversions result in fractional numbers. Those would get truncated through the system and be pretty much useless.

Public Member Functions

- [AAX_CUnitPrefixDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate)
- [AAX_CUnitPrefixDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateDecorator< T >](#)

- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate](#)< T > &displayDelegate)
Constructor.
- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegateDecorator](#) &other)
Copy constructor.
- [~AAX_IDisplayDelegateDecorator](#) () [AAX_OVERRIDE](#)
Virtual destructor.
- [AAX_IDisplayDelegateDecorator](#)< T > * [Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate decorator.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value with a size constraint.
- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.

- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.46.2 Constructor & Destructor Documentation**14.46.2.1 [AAX_CUnitPrefixDisplayDelegateDecorator](#)()**

```
template<typename T >
AAX_CUnitPrefixDisplayDelegateDecorator< T >::AAX_CUnitPrefixDisplayDelegateDecorator (
    const AAX\_IDisplayDelegate< T > & displayDelegate )
```

14.46.3 Member Function Documentation

14.46.3.1 Clone()

```
template<typename T >
AAX_CUnitPrefixDisplayDelegateDecorator< T > * AAX_CUnitPrefixDisplayDelegateDecorator< T >::
::Clone ( ) const [virtual]
```

Constructs and returns a copy of the display delegate.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>*      AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implements [AAX_IDisplayDelegate< T >](#).

14.46.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX_CUnitPrefixDisplayDelegateDecorator< T >::ValueToString (
    T value,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:



14.46.3.3 ValueToString() [2/2]

```
template<typename T >
bool AAX_CUnitPrefixDisplayDelegateDecorator< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_IDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the call graph for this function:

**14.46.3.4 StringToValue()**

```
template<typename T >
bool AAX_CUnitPrefixDisplayDelegateDecorator< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

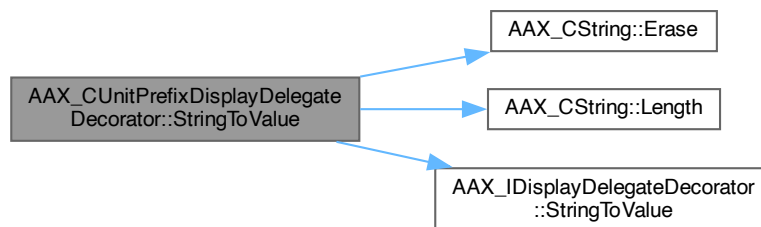
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

References [AAX_CString::Erase\(\)](#), [AAX_CString::Length\(\)](#), and [AAX_IDisplayDelegateDecorator< T >::StringToValue\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [AAX_CUnitPrefixDisplayDelegateDecorator.h](#)

14.47 AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT > Class Template Reference

```
#include <AAX_FastInterpolatedTableLookup.h>
```

Public Member Functions

- void [SetParameters](#) (int iTableSize, TFLOAT iMin=0.0, TFLOAT iMax=1.0, int iNumTables=1)
Set the table lookup parameters.
- DFLOAT [DoTableLookupExtraFast](#) (const TFLOAT *const iTable, DFLOAT iValue) const
Perform an extra fast table lookup :)
- void [DoTableLookupExtraFastMulti](#) (const TFLOAT *iTable, DFLOAT iValue, DFLOAT *oValues) const
- void [DoTableLookupExtraFast](#) (const TFLOAT *const iTable, const TFLOAT *const inpBuf, DFLOAT *const outBuf, int blockSize)
- TFLOAT [GetMin](#) ()
- TFLOAT [GetMaxMinusMin](#) ()

14.47.1 Member Function Documentation

14.47.1.1 SetParameters()

```
template<class TFLOAT , class DFLOAT >
void AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::SetParameters (
    int iTableSize,
    TFLOAT iMin = 0.0,
    TFLOAT iMax = 1.0,
    int iNumTables = 1 ) [inline]
```

Set the table lookup parameters.

Parameters

in	<i>iTableSize</i>	Size of the lookup table
in	<i>iMin</i>	Minimum input value
in	<i>iMax</i>	Maximum input value
in	<i>iNumTables</i>	Number of tables to index

Note

For future use...

14.47.1.2 DoTableLookupExtraFast() [1/2]

```
template<class TFLOAT , class DFLOAT >
DFLOAT AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::DoTableLookupExtraFast (
    const TFLOAT *const iTable,
    DFLOAT iValue ) const [inline]
```

Perform an extra fast table lookup :)

Parameters

in	<i>iTable</i>	Lookup table
in	<i>iValue</i>	Table value

Note

This version requires that the lookup table is padded with one extra location so we can avoid one of the checks to see if our pointers are out of bounds.

References [AAX::FastTrunc2Int32\(\)](#).

Here is the call graph for this function:



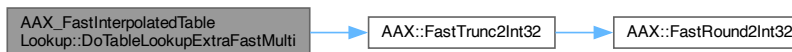
14.47.1.3 DoTableLookupExtraFastMulti()

```

template<class TFLOAT , class DFLOAT >
void AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::DoTableLookupExtraFastMulti (
    const TFLOAT * iTable,
    DFLOAT iValue,
    DFLOAT * oValues ) const [inline]
  
```

References [AAX::FastTrunc2Int32\(\)](#).

Here is the call graph for this function:



14.47.1.4 DoTableLookupExtraFast() [2/2]

```

template<class TFLOAT , class DFLOAT >
void AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::DoTableLookupExtraFast (
    const TFLOAT *const iTable,
    const TFLOAT *const inpBuf,
    DFLOAT *const outBuf,
    int blockSize ) [inline]
  
```

14.47.1.5 GetMin()

```

template<class TFLOAT , class DFLOAT >
TFLOAT AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::GetMin ( ) [inline]
  
```


14.47.1.6 GetMaxMinusMin()

```
template<class TFLOAT , class DFLOAT >  
TFLOAT AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >::GetMaxMinusMin ( ) [inline]
```

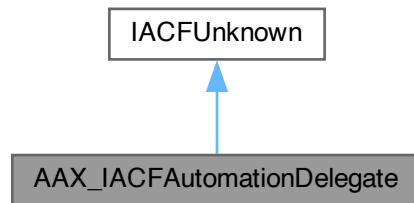
The documentation for this class was generated from the following files:

- [AAX_FastInterpolatedTableLookup.h](#)
- [AAX_FastInterpolatedTableLookup.hpp](#)

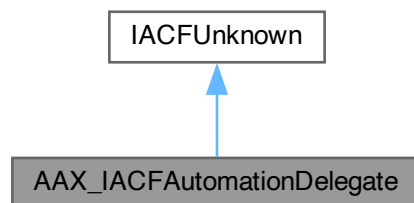
14.48 AAX_IACFAutomationDelegate Class Reference

```
#include <AAX_IACFAutomationDelegate.h>
```

Inheritance diagram for AAX_IACFAutomationDelegate:



Collaboration diagram for AAX_IACFAutomationDelegate:



14.48.1 Description

Versioned interface allowing an AAX plug-in to interact with the host's automation system.

See also

- [Parameter updates](#)
- [AAX_IAutomationDelegate](#)

Public Member Functions

- virtual [AAX_Result RegisterParameter](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result UnregisterParameter](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result PostSetValueRequest](#) ([AAX_CParamID](#) iParameterID, double normalizedValue) const =0
- virtual [AAX_Result PostCurrentValue](#) ([AAX_CParamID](#) iParameterID, double normalizedValue) const =0
- virtual [AAX_Result PostTouchRequest](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result PostReleaseRequest](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result GetTouchState](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *oTouched)=0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.48.2 Member Function Documentation

14.48.2.1 RegisterParameter()

```
virtual AAX\_Result AAX_IACFAutomationDelegate::RegisterParameter (
    AAX\_CParamID iParameterID ) [pure virtual]
```

Register a control with the automation system using a char* based control identifier

The automation delegate owns a list of the IDs of all of the parameters that have been registered with it. This list is used to set up listeners for all of the registered parameters such that the automation delegate may update the plug-in when the state of any of the registered parameters have been modified.

See also

[AAX_IAutomationDelegate::UnregisterParameter\(\)](#)

Parameters

in	<i>iParameterID</i>	Parameter ID that is being registered
----	---------------------	---------------------------------------

14.48.2.2 UnregisterParameter()

```
virtual AAX\_Result AAX_IACFAutomationDelegate::UnregisterParameter (
    AAX\_CParamID iParameterID ) [pure virtual]
```

Unregister a control with the automation system using a char* based control identifier

Note

All registered controls should be unregistered or the system might leak.

See also

[AAX_IAutomationDelegate::RegisterParameter\(\)](#)

Parameters

in	<i>iParameterID</i>	Parameter ID that is being registered
----	---------------------	---------------------------------------

14.48.2.3 PostSetValueRequest()

```
virtual AAX_Result AAX_IACFAutomationDelegate::PostSetValueRequest (
    AAX_CParamID iParameterID,
    double normalizedValue ) const [pure virtual]
```

Submits a request for the given parameter's value to be changed

Parameters

in	<i>iParameterID</i>	ID of the parameter for which a change is requested
in	<i>normalizedValue</i>	The requested new parameter value, formatted as a double and normalized to [0 1]

14.48.2.4 PostCurrentValue()

```
virtual AAX_Result AAX_IACFAutomationDelegate::PostCurrentValue (
    AAX_CParamID iParameterID,
    double normalizedValue ) const [pure virtual]
```

Notifies listeners that a parameter's value has changed

Parameters

in	<i>iParameterID</i>	ID of the parameter that has been updated
in	<i>normalizedValue</i>	The current parameter value, formatted as a double and normalized to [0 1]

14.48.2.5 PostTouchRequest()

```
virtual AAX_Result AAX_IACFAutomationDelegate::PostTouchRequest (
    AAX_CParamID iParameterID ) [pure virtual]
```

Requests that the given parameter be "touched", i.e. locked for updates by the current client

Parameters

in	<i>iParameterID</i>	ID of the parameter that will be touched
----	---------------------	--

14.48.2.6 PostReleaseRequest()

```
virtual AAX_Result AAX_IACFAutomationDelegate::PostReleaseRequest (
    AAX_CParamID iParameterID ) [pure virtual]
```

Requests that the given parameter be "released", i.e. available for updates from any client

Parameters

in	<i>iParameterID</i>	ID of the parameter that will be released
----	---------------------	---

14.48.2.7 GetTouchState()

```
virtual AAX_Result AAX_IACFAutomationDelegate::GetTouchState (
    AAX_CParamID iParameterID,
    AAX_CBoolean * oTouched ) [pure virtual]
```

Gets the current touched state of a parameter

Parameters

in	<i>iParameterID</i>	ID of the parameter that is being queried
out	<i>oTouched</i>	The current touch state of the parameter

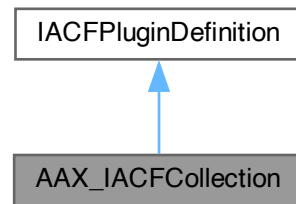
The documentation for this class was generated from the following file:

- [AAX_IACFAutomationDelegate.h](#)

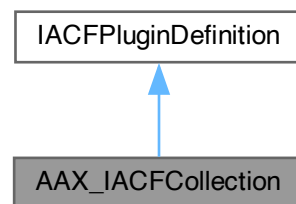
14.49 AAX_IACFCollection Class Reference

```
#include <AAX_IACFCollection.h>
```

Inheritance diagram for AAX_IACFCollection:



Collaboration diagram for AAX_IACFCollection:



14.49.1 Description

Versioned interface to represent a plug-in binary's static description.

Public Member Functions

- virtual [AAX_Result AddEffect](#) (const char *inEffectID, [IACFUnknown](#) *inEffectDescriptor)=0
Add an Effect description to the collection.
- virtual [AAX_Result SetManufacturerName](#) (const char *inPackageName)=0
Set the plug-in manufacturer name.
- virtual [AAX_Result AddPackageName](#) (const char *inPackageName)=0
Set the plug-in package name.
- virtual [AAX_Result SetPackageVersion](#) (uint32_t inVersion)=0
Set the plug-in package version number.
- virtual [AAX_Result SetProperties](#) ([IACFUnknown](#) *inProperties)=0
Set the properties of the collection.

14.49.2 Member Function Documentation

14.49.2.1 AddEffect()

```
virtual AAX_Result AAX_IACFCollection::AddEffect (
    const char * inEffectID,
    IACFUnknown * inEffectDescriptor ) [pure virtual]
```

Add an Effect description to the collection.

Each Effect that a plug-in registers with [AAX_ICollection::AddEffect\(\)](#) is considered a completely different user-facing product. For example, in Avid's Dynamics III plug-in the Expander, Compressor, and DeEsser are each registered as separate Effects. All stem format variations within each Effect are registered within that Effect's [AAX_IEffectDescriptor](#) using [AddComponent\(\)](#).

The [AAX_eProperty_ProductID](#) value for all ProcessProcs within a single Effect must be identical.

This method passes ownership of an [AAX_IEffectDescriptor](#) object to the [AAX_ICollection](#). The [AAX_IEffectDescriptor](#) must not be deleted by the AAX plug-in, nor should it be edited in any way after it is passed to the [AAX_ICollection](#).

Parameters

in	<i>inEffectID</i>	The effect ID.
in	<i>inEffectDescriptor</i>	The Effect descriptor.

14.49.2.2 SetManufacturerName()

```
virtual AAX_Result AAX_IACFCollection::SetManufacturerName (
    const char * inPackageName ) [pure virtual]
```

Set the plug-in manufacturer name.

Parameters

in	<i>inPackageName</i>	The name of the manufacturer.
----	----------------------	-------------------------------

14.49.2.3 AddPackageName()

```
virtual AAX_Result AAX_IACFCollection::AddPackageName (
    const char * inPackageName ) [pure virtual]
```

Set the plug-in package name.

May be called multiple times to add abbreviated package names.

Note

Every plug-in must include at least one name variant with 16 or fewer characters, plus a null terminating character. Used for Presets folder.

Parameters

in	<i>inPackageName</i>	The name of the package.
----	----------------------	--------------------------

14.49.2.4 SetPackageVersion()

```
virtual AAX_Result AAX_IACFCollection::SetPackageVersion (
    uint32_t inVersion ) [pure virtual]
```

Set the plug-in package version number.

Parameters

in	<i>inVersion</i>	The package version numner.
----	------------------	-----------------------------

14.49.2.5 SetPropertyies()

```
virtual AAX_Result AAX_IACFCollection::SetProperties (
    IACFUnknown * inProperties ) [pure virtual]
```

Set the properties of the collection.

Parameters

in	<i>inProperties</i>	Collection properties
----	---------------------	-----------------------

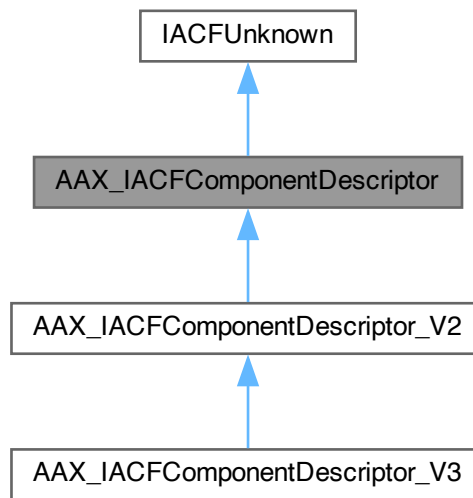
The documentation for this class was generated from the following file:

- [AAX_IACFCollection.h](#)

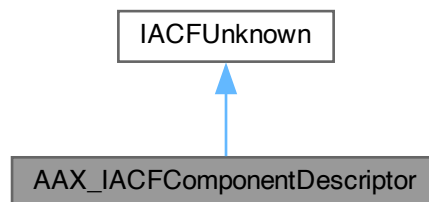
14.50 AAX_IACFComponentDescriptor Class Reference

```
#include <AAX_IACFComponentDescriptor.h>
```

Inheritance diagram for AAX_IACFComponentDescriptor:



Collaboration diagram for AAX_IACFComponentDescriptor:



14.50.1 Description

Versioned description interface for an AAX plug-in algorithm callback.

Public Member Functions

- virtual [AAX_Result Clear](#) ()=0
Clears the descriptor.
- virtual [AAX_Result AddReservedField](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inFieldType)=0
Subscribes a context field to host-provided services or information.

- virtual [AAX_Result AddAudioIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio input context field.
- virtual [AAX_Result AddAudioOut](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio output context field.
- virtual [AAX_Result AddAudioBufferLength](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a buffer length context field.
- virtual [AAX_Result AddSampleRate](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a sample rate context field.
- virtual [AAX_Result AddClock](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a clock context field.
- virtual [AAX_Result AddSideChainIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a side-chain input context field.
- virtual [AAX_Result AddDataInPort](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inPacketSize, [AAX_EDataInPortType](#) inPortType)=0
Adds a custom data port to the algorithm context.
- virtual [AAX_Result AddAuxOutputStem](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inStemFormat, [const char](#) inNameUTF8[])=0
Adds an auxiliary output stem for a plug-in.
- virtual [AAX_Result AddPrivateData](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inDataSize, [uint32_t](#) inOptions=[AAX_ePrivateDataOptions_DefaultOptions](#))=0
Adds a private data port to the algorithm context.
- virtual [AAX_Result AddDmaInstance](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_IDma::EMode](#) inDmaMode)=0
Adds a DMA field to the plug-in's context.
- virtual [AAX_Result AddMIDINode](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_EMIDINodeType](#) inNodeType, [const char](#) inNodeName[], [uint32_t](#) channelMask)=0
Adds a MIDI node field to the plug-in's context.
- virtual [AAX_Result AddProcessProc_Native](#) ([AAX_CProcessProc](#) inProcessProc, [IACFUnknown](#) *inProperties, [AAX_CInstanceInitProc](#) inInstanceInitProc, [AAX_CBackgroundProc](#) inBackgroundProc, [AAX_CSelector](#) *outProcID)=0
Registers an algorithm processing endpoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc_TI](#) ([const char](#) inDLLFileNameUTF8[], [const char](#) inProcessProcSymbol[], [IACFUnknown](#) *inProperties, [const char](#) inInstanceInitProcSymbol[], [const char](#) inBackgroundProcSymbol[], [AAX_CSelector](#) *outProcID)=0
Registers an algorithm processing endpoint (process procedure) for the native architecture.
- virtual [AAX_Result AddMeters](#) ([AAX_CFieldIndex](#) inFieldIndex, [const AAX_CTypeID](#) *inMeterIDs, [const uint32_t](#) inMeterCount)=0
Adds a meter field to the plug-in's context.

Public Member Functions inherited from [IACFUnknown](#)

- virtual [BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE QueryInterface](#) ([const acfIID](#) &iid, [void **ppOut](#))=0
Returns pointers to supported interfaces.
- virtual [acfUInt32 ACFMETHODCALLTYPE AddRef](#) ([void](#))=0
Increments reference count.
- virtual [acfUInt32 ACFMETHODCALLTYPE Release](#) ([void](#))=0
Decrements reference count.

14.50.2 Member Function Documentation

14.50.2.1 Clear()

```
virtual AAX_Result AAX_IACFComponentDescriptor::Clear ( ) [pure virtual]
```

Clears the descriptor.

Clears the descriptor and readies it for the next algorithm description

14.50.2.2 AddReservedField()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddReservedField (
    AAX_CFieldIndex inFieldIndex,
    uint32_t inFieldType ) [pure virtual]
```

Subscribes a context field to host-provided services or information.

Note

Currently for internal use only.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inFieldType</i>	Type of field that is being added

14.50.2.3 AddAudioIn()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddAudioIn (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes an audio input context field.

Defines an audio in port for host-provided information in the algorithm's context structure.

- Data type: float**
- Data kind: An array of float arrays, one for each input channel

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

14.50.2.4 AddAudioOut()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddAudioOut (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes an audio output context field.

Defines an audio out port for host-provided information in the algorithm's context structure.

- Data type: float**
- Data kind: An array of float arrays, one for each output channel

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

14.50.2.5 AddAudioBufferLength()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddAudioBufferLength (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a buffer length context field.

Defines a buffer length port for host-provided information in the algorithm's context structure.

- Data type: int32_t*
- Data kind: The number of samples in the current audio buffer

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

14.50.2.6 AddSampleRate()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddSampleRate (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a sample rate context field.

Defines a sample rate port for host-provided information in the algorithm's context structure.

- Data type: [AAX_CSampleRate](#) *
- Data kind: The current sample rate

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

14.50.2.7 AddClock()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddClock (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a clock context field.

Defines a clock port for host-provided information in the algorithm's context structure.

- Data type: [AAX_CTimestamp](#) *
- Data kind: A running counter which increments even when the transport is not playing. The counter increments exactly once per sample quantum.

Host Compatibility Notes As of Pro Tools 11.1, this field may be used in both Native and DSP plug-ins. The DSP clock data is a 16-bit cycling counter. This field was only available for Native plug-ins in previous Pro Tools versions.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

14.50.2.8 AddSideChainIn()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddSideChainIn (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a side-chain input context field.

Defines a side-chain input port for host-provided information in the algorithm's context structure.

- Data type: `int32_t*`
- Data kind: The index of the plug-in's first side-chain input channel within the array of input audio buffers

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

14.50.2.9 AddDataInPort()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddDataInPort (
    AAX_CFieldIndex inFieldIndex,
    uint32_t inPacketSize,
    AAX_EDataInPortType inPortType ) [pure virtual]
```

Adds a custom data port to the algorithm context.

Defines a read-only data port for plug-in information in the algorithm's context structure. The plug-in can send information to this port using [AAX_IController::PostPacket\(\)](#).

The host guarantees that all packets will be delivered to this port in the order in which they were posted, up to the point of a packet buffer overflow, though some packets may be dropped depending on the `inPortType` and host implementation.

Note

When a plug-in is operating in offline (AudioSuite) mode, all data ports operate as [AAX_eDataInPortType_Unbuffered](#) ports

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inPacketSize</i>	Size of the data packets that will be sent to this port
in	<i>inPortType</i>	The requested packet delivery behavior for this port

14.50.2.10 AddAuxOutputStem()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddAuxOutputStem (
    AAX_CFieldIndex inFieldIndex,
    int32_t inStemFormat,
    const char inNameUTF8[] ) [pure virtual]
```

Adds an auxiliary output stem for a plug-in.

Use this method to add additional output channels to the algorithm context.

The aux output stem audio buffers will be added to the end of the audio outputs array in the order in which they are described. When writing audio data to a specific aux output, find the proper starting channel by accumulating all of the channels of the main output stem format and any previously-described aux output stems.

The plug-in is responsible for providing a meaningful name for each aux outputs. At the very least, individual outputs should be labeled "Output xx", where "xx" is the aux output number as it is defined in the plug-in. The output name should also include the words "mono" and "stereo" to support when users are looking for an output with a specific stem format.

Host Compatibility Notes There is a hard limit to the number of outputs that Pro Tools supports for a single plug-in instance. This limit is currently set at 256 channels, which includes all of the plug-in's output channels in addition to the sum total of all of its aux output stem channels.

Host Compatibility Notes Pro Tools supports only mono and stereo auxiliary output stem formats

Warning

This method will return an error code on hosts which do not support auxiliary output stems. This indicates that the host will not provide audio buffers for auxiliary output stems during processing. A plug-in must not attempt to write data into auxiliary output stem buffers which have not been provided by the host!

Parameters

in	<i>inFieldIndex</i>	DEPRECATED: This parameter is no longer needed by the host, but is included in the interface for binary compatibility
in	<i>inStemFormat</i>	The stem format of the new aux output
in	<i>inNameUTF8</i>	The name of the aux output. This name is static and cannot be changed after the descriptor is submitted to the host

14.50.2.11 AddPrivateData()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddPrivateData (
    AAX_CFieldIndex inFieldIndex,
    int32_t inDataSize,
    uint32_t inOptions = AAX_ePrivateDataOptions_DefaultOptions ) [pure virtual]
```

Adds a private data port to the algorithm context.

Defines a read/write data port for private state data. Data written to this port will be maintained by the host between calls to the algorithm context.

See also

alg_pd_registration

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inDataSize</i>	Size of the data packets that will be sent to this port
in	<i>inOptions</i>	Options that define the private data port's behavior

14.50.2.12 AddDmaInstance()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddDmaInstance (
    AAX_CFieldIndex inFieldIndex,
    AAX_IDma::EMode inDmaMode ) [pure virtual]
```

Adds a DMA field to the plug-in's context.

DMA (direct memory access) provides efficient reads from and writes to external memory on the DSP. DMA behavior is emulated in host-based plug-ins for cross-platform portability.

Note

The order in which DMA instances are added defines their priority and therefore order of execution of DMA operations. In most plug-ins, Scatter fields should be placed first in order to achieve the lowest possible access latency.

For more information, see [Direct Memory Access](#) .

Todo Update the DMA system management such that operation priority can be set arbitrarily

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inDmaMode</i>	AAX_IDma::EMode that will apply to this field

14.50.2.13 AddMIDINode()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddMIDINode (
    AAX_CFieldIndex inFieldIndex,
    AAX_EMIDINodeType inNodeType,
    const char inNodeName[],
    uint32_t channelMask ) [pure virtual]
```

Adds a MIDI node field to the plug-in's context.

- Data type: [AAX_IMIDINode](#) *

The resulting MIDI node data will be available both in the algorithm context and in the plug-in's [data model](#) via [UpdateMIDINodes\(\)](#).

To add a MIDI node that is only accessible to the plug-in's data model, use [AAX_IEffectDescriptor::AddControlMIDINode\(\)](#)

Host Compatibility Notes Due to current restrictions MIDI data won't be delivered to DSP algorithms, only to AAX Native.

Parameters

in	<i>inFieldIndex</i>	The ID of the port. MIDI node ports should formatted as a pointer to an AAX_IMIDINode .
in	<i>inNodeType</i>	The type of MIDI node, as AAX_EMIDINodeType
in	<i>inNodeName</i>	The name of the MIDI node as it should appear in the host's UI
in	<i>channelMask</i>	The channel mask for the MIDI node. This parameter specifies used MIDI channels. For Global MIDI nodes, use a mask of AAX_EMidiGlobalNodeSelectors

14.50.2.14 AddProcessProc_Native()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddProcessProc_Native (
    AAX_CProcessProc inProcessProc,
    IACFUnknown * inProperties,
    AAX_CInstanceInitProc inInstanceInitProc,
    AAX_CBackgroundProc inBackgroundProc,
    AAX_CSelector * outProcID ) [pure virtual]
```

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

Parameters

in	<i>inProcessProc</i>	Symbol for this processing callback
in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
in	<i>inInstanceInitProc</i>	Initialization routine that will be called when a new instance of the Effect is created. See Algorithm initialization .
in	<i>inBackgroundProc</i>	Background routine that will be called in an idle context within the same address space as the associated process procedure. See Background processing callback
out	<i>outProcID</i>	

Todo document this parameter

14.50.2.15 AddProcessProc_TI()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddProcessProc_TI (
    const char inDLLFileNameUTF8[],
    const char inProcessProcSymbol[],
    IACFUnknown * inProperties,
    const char inInstanceInitProcSymbol[],
    const char inBackgroundProcSymbol[],
    AAX_CSelector * outProcID ) [pure virtual]
```

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

Parameters

in	<i>inDLLFileNameUTF8</i>	UTF-8 encoded filename for the ELF DLL containing the algorithm code fragment
in	<i>inProcessProcSymbol</i>	Symbol for this processing callback
in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
in	<i>inInstanceInitProcSymbol</i>	Initialization routine that will be called when a new instance of the Effect is created. Must be included in the same DLL as the main algorithm entrypoint. See Algorithm initialization .

Parameters

in	<i>inBackgroundProcSymbol</i>	Background routine that will be called in an idle context within the same address space as the associated process procedure. Must be included in the same DLL as the main algorithm entrypoint. See Background processing callback
out	<i>outProcID</i>	

Todo document this parameter

14.50.2.16 AddMeters()

```
virtual AAX_Result AAX_IACFComponentDescriptor::AddMeters (
    AAX_CFieldIndex inFieldIndex,
    const AAX_CTypeID * inMeterIDs,
    const uint32_t inMeterCount ) [pure virtual]
```

Adds a meter field to the plug-in's context.

Meter fields include an array of meter tap values, with one tap per meter per context. Only one meter field should be added per Component. Individual meter behaviors can be described at the Effect level.

For more information, see [Plug-in meters](#) .

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inMeterIDs</i>	Array of 32-bit IDs, one for each meter. Meter IDs must be unique within the Effect.
in	<i>inMeterCount</i>	The number of meters included in this field

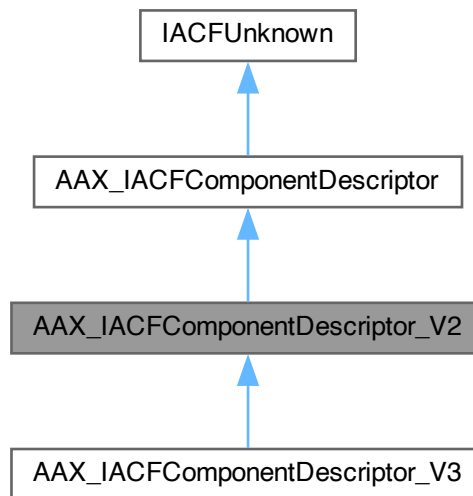
The documentation for this class was generated from the following file:

- [AAX_IACFComponentDescriptor.h](#)

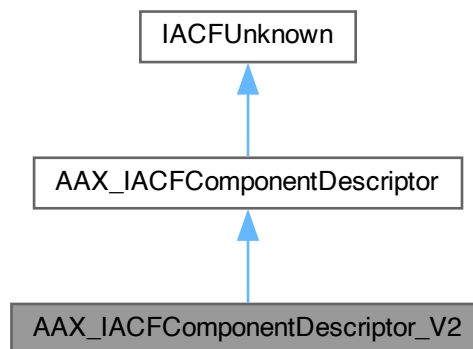
14.51 AAX_IACFComponentDescriptor_V2 Class Reference

```
#include <AAX_IACFComponentDescriptor.h>
```

Inheritance diagram for AAX_IACFComponentDescriptor_V2:



Collaboration diagram for AAX_IACFComponentDescriptor_V2:



14.51.1 Description

Versioned description interface for an AAX plug-in algorithm callback.

Public Member Functions

- virtual [AAX_Result AddTemporaryData](#) ([AAX_CFieldIndex](#) inFieldIndex, uint32_t inDataElementSize)=0
Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.

Public Member Functions inherited from [AAX_IACFComponentDescriptor](#)

- virtual [AAX_Result Clear](#) ()=0
Clears the descriptor.
- virtual [AAX_Result AddReservedField](#) ([AAX_CFieldIndex](#) inFieldIndex, uint32_t inFieldType)=0
Subscribes a context field to host-provided services or information.
- virtual [AAX_Result AddAudioIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio input context field.
- virtual [AAX_Result AddAudioOut](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio output context field.
- virtual [AAX_Result AddAudioBufferLength](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a buffer length context field.
- virtual [AAX_Result AddSampleRate](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a sample rate context field.
- virtual [AAX_Result AddClock](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a clock context field.
- virtual [AAX_Result AddSideChainIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a side-chain input context field.
- virtual [AAX_Result AddDataInPort](#) ([AAX_CFieldIndex](#) inFieldIndex, uint32_t inPacketSize, [AAX_EDataInPortType](#) inPortType)=0
Adds a custom data port to the algorithm context.
- virtual [AAX_Result AddAuxOutputStem](#) ([AAX_CFieldIndex](#) inFieldIndex, int32_t inStemFormat, const char inNameUTF8[])=0
Adds an auxiliary output stem for a plug-in.
- virtual [AAX_Result AddPrivateData](#) ([AAX_CFieldIndex](#) inFieldIndex, int32_t inDataSize, uint32_t inOptions=[AAX_ePrivateDataOptions_DefaultOptions](#))=0
Adds a private data port to the algorithm context.
- virtual [AAX_Result AddDmaInstance](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_IDma::EMode](#) inDmaMode)=0
Adds a DMA field to the plug-in's context.
- virtual [AAX_Result AddMIDINode](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_EMIDINodeType](#) inNodeType, const char inNodeName[], uint32_t channelMask)=0
Adds a MIDI node field to the plug-in's context.
- virtual [AAX_Result AddProcessProc_Native](#) ([AAX_CProcessProc](#) inProcessProc, [IACFUnknown](#) *inProperties, [AAX_CInstanceInitProc](#) inInstanceInitProc, [AAX_CBackgroundProc](#) inBackgroundProc, [AAX_CSelector](#) *outProcID)=0
Registers an algorithm processing endpoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc_TI](#) (const char inDLLFileNameUTF8[], const char inProcessProcSymbol[], [IACFUnknown](#) *inProperties, const char inInstanceInitProcSymbol[], const char inBackgroundProcSymbol[], [AAX_CSelector](#) *outProcID)=0
Registers an algorithm processing endpoint (process procedure) for the native architecture.
- virtual [AAX_Result AddMeters](#) ([AAX_CFieldIndex](#) inFieldIndex, const [AAX_CTypeID](#) *inMeterIDs, const uint32_t inMeterCount)=0
Adds a meter field to the plug-in's context.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.51.2 Member Function Documentation

14.51.2.1 AddTemporaryData()

```
virtual AAX_Result AAX_IACFComponentDescriptor_V2::AddTemporaryData (
    AAX_CFieldIndex inFieldIndex,
    uint32_t inDataElementSize ) [pure virtual]
```

Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.

This can be very useful if you use block processing and need to store intermediate results. Just specify your base element size and the system will scale the overall block size by the buffer size. For example, to create a buffer of floats that is the length of the block, specify 4 bytes as the elementsize.

This data block does not retain state across callback and can also be reused across instances on memory constrained systems.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inDataElementSize</i>	The size of a single piece of data in the block. This number will be multiplied by the processing block size to determine total block size.

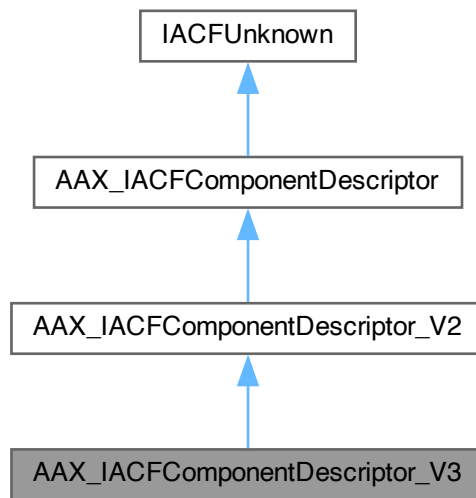
The documentation for this class was generated from the following file:

- [AAX_IACFComponentDescriptor.h](#)

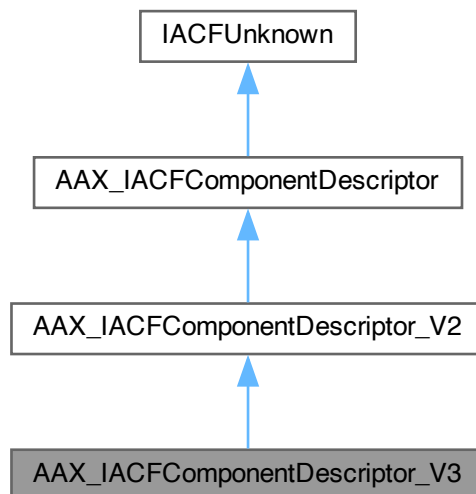
14.52 AAX_IACFComponentDescriptor_V3 Class Reference

```
#include <AAX_IACFComponentDescriptor.h>
```

Inheritance diagram for AAX_IACFComponentDescriptor_V3:



Collaboration diagram for AAX_IACFComponentDescriptor_V3:



14.52.1 Description

Versioned description interface for an AAX plug-in algorithm callback.

Public Member Functions

- virtual [AAX_Result AddProcessProc](#) ([IACFUnknown](#) *inProperties, [AAX_CSelector](#) *outProcIDs, [int32_t](#) inProcIDsSize)=0
Registers one or more algorithm processing entrypoints (process procedures)

Public Member Functions inherited from [AAX_IACFComponentDescriptor_V2](#)

- virtual [AAX_Result AddTemporaryData](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inDataElementSize)=0
Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.

Public Member Functions inherited from [AAX_IACFComponentDescriptor](#)

- virtual [AAX_Result Clear](#) ()=0
Clears the descriptor.
- virtual [AAX_Result AddReservedField](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inFieldType)=0
Subscribes a context field to host-provided services or information.
- virtual [AAX_Result AddAudioIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio input context field.
- virtual [AAX_Result AddAudioOut](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio output context field.
- virtual [AAX_Result AddAudioBufferLength](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a buffer length context field.
- virtual [AAX_Result AddSampleRate](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a sample rate context field.
- virtual [AAX_Result AddClock](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a clock context field.
- virtual [AAX_Result AddSideChainIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a side-chain input context field.
- virtual [AAX_Result AddDataInPort](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inPacketSize, [AAX_EDataInPortType](#) inPortType)=0
Adds a custom data port to the algorithm context.
- virtual [AAX_Result AddAuxOutputStem](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inStemFormat, [const char](#) inNameUTF8[])=0
Adds an auxiliary output stem for a plug-in.
- virtual [AAX_Result AddPrivateData](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inDataSize, [uint32_t](#) inOptions=[AAX_ePrivateDataOptions_DefaultOptions](#))=0
Adds a private data port to the algorithm context.
- virtual [AAX_Result AddDmaInstance](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_IDma::EMode](#) inDmaMode)=0
Adds a DMA field to the plug-in's context.
- virtual [AAX_Result AddMIDINode](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_EMIDINodeType](#) inNodeType, [const char](#) inNodeName[], [uint32_t](#) channelMask)=0
Adds a MIDI node field to the plug-in's context.
- virtual [AAX_Result AddProcessProc_Native](#) ([AAX_CProcessProc](#) inProcessProc, [IACFUnknown](#) *inProperties, [AAX_CInstanceInitProc](#) inInstanceInitProc, [AAX_CBackgroundProc](#) inBackgroundProc, [AAX_CSelector](#) *outProcID)=0
Registers an algorithm processing entrypoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc_TI](#) ([const char](#) inDLLFileNameUTF8[], [const char](#) inProcessProcSymbol[], [IACFUnknown](#) *inProperties, [const char](#) inInstanceInitProcSymbol[], [const char](#) inBackgroundProcSymbol[], [AAX_CSelector](#) *outProcID)=0
Registers an algorithm processing entrypoint (process procedure) for the native architecture.
- virtual [AAX_Result AddMeters](#) ([AAX_CFieldIndex](#) inFieldIndex, [const AAX_CTypeID](#) *inMeterIDs, [const uint32_t](#) inMeterCount)=0
Adds a meter field to the plug-in's context.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.52.2 Member Function Documentation**14.52.2.1 AddProcessProc()**

```
virtual AAX\_Result AAX_IACFComponentDescriptor_V3::AddProcessProc (
    IACFUnknown * inProperties,
    AAX\_CSelector * outProcIDs,
    int32_t inProcIDsSize ) [pure virtual]
```

Registers one or more algorithm processing entrypoints (process procedures)

Any non-overlapping set of processing entrypoints may be specified. Typically this can be used to specify both Native and TI entrypoints using the same call.

The AAX Library implementation of this method includes backwards compatibility logic to complete the ProcessProc registration on hosts which do not support this method. Therefore plug-in code may use this single registration routine instead of separate calls to [AddProcessProc_Native\(\)](#), [AddProcessProc_TI\(\)](#), etc. regardless of the host version.

The following properties replace the input arguments to the platform-specific registration methods:

[AddProcessProc_Native\(\)](#) ([AAX_eProperty_PluginID_Native](#), [AAX_eProperty_PluginID_AudioSuite](#))

- [AAX_CProcessProc](#) iProcessProc: [AAX_eProperty_NativeProcessProc](#) (required)
- [AAX_CInstanceInitProc](#) iInstanceInitProc: [AAX_eProperty_NativeInstanceInitProc](#) (optional)
- [AAX_CBackgroundProc](#) iBackgroundProc: [AAX_eProperty_NativeBackgroundProc](#) (optional)

[AddProcessProc_TI\(\)](#) ([AAX_eProperty_PluginID_TI](#))

- const char inDLLFileNameUTF8[]: [AAX_eProperty_TIDLLFileName](#) (required)
- const char iProcessProcSymbol[]: [AAX_eProperty_TIPProcessProc](#) (required)
- const char iInstanceInitProcSymbol[]: [AAX_eProperty_TIInstanceInitProc](#) (optional)
- const char iBackgroundProcSymbol[]: [AAX_eProperty_TIBackgroundProc](#) (optional)

If any platform-specific plug-in ID property is present in iProperties then [AddProcessProc\(\)](#) will check for the required properties for that platform.

Note

[AAX_eProperty_AudioBufferLength](#) will be ignored for the Native and AudioSuite ProcessProcs since it should only be used for AAX DSP.

Parameters

in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
out	<i>outProcIDs</i>	

Todo document this parameter Returned array will be NULL-terminated

Parameters

in	<i>inProcIDsSize</i>	The size of the array provided to oProcIDs. If oProcIDs is non-NULL but iProcIDsSize is not large enough for all of the registered ProcessProcs (plus one for NULL termination) then this method will fail with AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW
----	----------------------	--

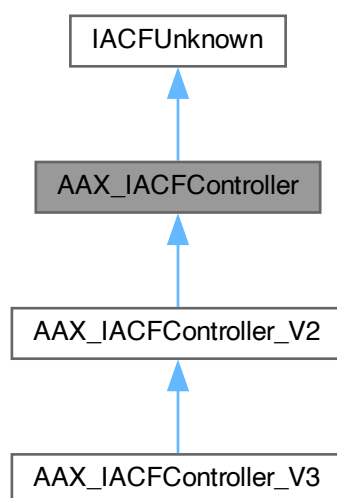
The documentation for this class was generated from the following file:

- [AAX_IACFComponentDescriptor.h](#)

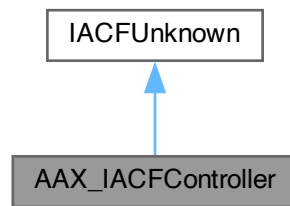
14.53 AAX_IACFController Class Reference

```
#include <AAX_IACFController.h>
```

Inheritance diagram for AAX_IACFController:



Collaboration diagram for AAX_IACFController:



14.53.1 Description

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.

Public Member Functions

- virtual [AAX_Result GetEffectID](#) ([AAX_IString](#) *outEffectID) const =0
- virtual [AAX_Result GetSampleRate](#) ([AAX_CSampleRate](#) *outSampleRate) const =0
CALL: Returns the current literal sample rate.
- virtual [AAX_Result GetInputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's input stem format.
- virtual [AAX_Result GetOutputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's output stem format.
- virtual [AAX_Result GetSignalLatency](#) (int32_t *outSamples) const =0
CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.
- virtual [AAX_Result GetCycleCount](#) ([AAX_EProperty](#) inWhichCycleCount, [AAX_CPropertyValue](#) *outNumCycles) const =0
CALL: returns the plug-in's current real-time DSP cycle count.
- virtual [AAX_Result GetTODLocation](#) ([AAX_CTimeOfDay](#) *outTODLocation) const =0
CALL: Returns the current Time Of Day (TOD) of the system.
- virtual [AAX_Result SetSignalLatency](#) (int32_t inNumSamples)=0
CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.
- virtual [AAX_Result SetCycleCount](#) ([AAX_EProperty](#) *inWhichCycleCounts, [AAX_CPropertyValue](#) *iValues, int32_t numValues)=0
CALL: Indicates a change in the plug-in's real-time DSP cycle count.
- virtual [AAX_Result PostPacket](#) ([AAX_CFieldIndex](#) inFieldIndex, const void *inPayloadP, uint32_t inPayloadSize)=0
CALL: Posts a data packet to the host for routing between plug-in components.
- virtual [AAX_Result GetCurrentMeterValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterValue) const =0
CALL: Retrieves the current value of a host-managed plug-in meter.
- virtual [AAX_Result GetMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterPeakValue) const =0
CALL: Retrieves the currently held peak value of a host-managed plug-in meter.
- virtual [AAX_Result ClearMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID) const =0

- CALL: Clears the peak value from a host-managed plug-in meter.*
- virtual [AAX_Result GetMeterClipped](#) ([AAX_CTypeID](#) inMeterID, [AAX_CBoolean](#) *outClipped) const =0
CALL: Retrieves the clipped flag from a host-managed plug-in meter.
- virtual [AAX_Result ClearMeterClipped](#) ([AAX_CTypeID](#) inMeterID) const =0
CALL: Clears the clipped flag from a host-managed plug-in meter.
- virtual [AAX_Result GetMeterCount](#) (uint32_t *outMeterCount) const =0
CALL: Retrieves the number of host-managed meters registered by a plug-in.
- virtual [AAX_Result GetNextMIDIPacket](#) ([AAX_CFieldIndex](#) *outPort, [AAX_CMidiPacket](#) *outPacket)=0
CALL: Retrieves MIDI packets for described MIDI nodes.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.53.2 Member Function Documentation

14.53.2.1 GetEffectID()

```
virtual AAX\_Result AAX_IACFController::GetEffectID (
    AAX\_IString * outEffectID ) const [pure virtual]
```

14.53.2.2 GetSampleRate()

```
virtual AAX\_Result AAX_IACFController::GetSampleRate (
    AAX\_CSampleRate * outSampleRate ) const [pure virtual]
```

CALL: Returns the current literal sample rate.

Parameters

out	<i>outSampleRate</i>	The current sample rate
-----	----------------------	-------------------------

14.53.2.3 GetInputStemFormat()

```
virtual AAX\_Result AAX_IACFController::GetInputStemFormat (
```

```
AAX_EStemFormat * outStemFormat ) const [pure virtual]
```

CALL: Returns the plug-in's input stem format.

Parameters

out	<i>outStemFormat</i>	The current input stem format
-----	----------------------	-------------------------------

14.53.2.4 GetOutputStemFormat()

```
virtual AAX_Result AAX_IACFController::GetOutputStemFormat (
    AAX_EStemFormat * outStemFormat ) const [pure virtual]
```

CALL: Returns the plug-in's output stem format.

Parameters

out	<i>outStemFormat</i>	The current output stem format
-----	----------------------	--------------------------------

14.53.2.5 GetSignalLatency()

```
virtual AAX_Result AAX_IACFController::GetSignalLatency (
    int32_t * outSamples ) const [pure virtual]
```

CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.

This method provides the most recently published signal latency. The host may not have updated its delay compensation to match this signal latency yet, so plug-ins that dynamically change their latency using [SetSignalLatency\(\)](#) should always wait for an [AAX_eNotificationEvent_SignalLatencyChanged](#) notification before updating its algorithm to incur this latency.

See also

[SetSignalLatency\(\)](#)

Parameters

out	<i>outSamples</i>	The number of samples of signal delay published by the plug-in
-----	-------------------	--

14.53.2.6 GetCycleCount()

```
virtual AAX_Result AAX_IACFController::GetCycleCount (
```

```
AAX_EProperty inWhichCycleCount,
AAX_CPropertyValue * outNumCycles ) const [pure virtual]
```

CALL: returns the plug-in's current real-time DSP cycle count.

This method provides the number of cycles that the AAX host expects the DSP plug-in to consume. The host uses this value when allocating DSP resources for the plug-in.

Note

A plug-in should never apply a DSP algorithm with more demanding resource requirements than what is currently accounted for by the host. To set a higher cycle count value, a plug-in must call [AAX_IController::SetCycleCount\(\)](#), then poll [AAX_IController::GetCycleCount\(\)](#) until the new value has been applied. Once the host has recognized the new cycle count value, the plug-in may apply the more demanding algorithm.

Parameters

in	<i>inWhichCycleCount</i>	Selector for the requested cycle count metric. One of: <ul style="list-style-type: none"> • AAX_eProperty_TI_SharedCycleCount • AAX_eProperty_TI_InstanceCycleCount • AAX_eProperty_TI_MaxInstancesPerChip
in	<i>outNumCycles</i>	The current value of the selected cycle count metric

Todo PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

14.53.2.7 GetTODLocation()

```
virtual AAX_Result AAX_IACFController::GetTODLocation (
    AAX_CTimeOfDay * outTODLocation ) const [pure virtual]
```

CALL: Returns the current Time Of Day (TOD) of the system.

This method provides a plug-in the TOD (in samples) of the current system. TOD is the number of samples that the playhead has traversed since the beginning of playback.

Note

The TOD value is the immediate value of the audio engine playhead. This value is incremented within the audio engine's real-time rendering context; it is not synchronized with non-real-time calls to plug-in interface methods.

Parameters

out	<i>outTODLocation</i>	The current Time Of Day as set by the host
-----	-----------------------	--

14.53.2.8 SetSignalLatency()

```
virtual AAX_Result AAX_IACFController::SetSignalLatency (
    int32_t inNumSamples ) [pure virtual]
```

CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.

This method is used to request a change in the number of samples that the AAX host expects the plug-in to delay a signal.

The host is not guaranteed to immediately apply the new latency value. A plug-in should avoid incurring an actual algorithmic latency that is different than the latency accounted for by the host.

To set a new latency value, a plug-in must call [AAX_IController::SetSignalLatency\(\)](#), then wait for an [AAX_eNotificationEvent_SignalLatencyChanged](#) notification. Once this notification has been received, [AAX_IController::GetSignalLatency\(\)](#) will reflect the updated latency value and the plug-in should immediately apply any relevant algorithmic changes that alter its latency to this new value.

Warning

Parameters which affect the latency of a plug-in should not be made available for control through automation. This will result in audible glitches when delay compensation is adjusted while playing back automation for these parameters.

Parameters

in	<i>inNumSamples</i>	The number of samples of signal delay that the plug-in requests to incur
----	---------------------	--

14.53.2.9 SetCycleCount()

```
virtual AAX_Result AAX_IACFController::SetCycleCount (
    AAX_EProperty * inWhichCycleCounts,
    AAX_CPropertyValue * iValues,
    int32_t numValues ) [pure virtual]
```

CALL: Indicates a change in the plug-in's real-time DSP cycle count.

This method is used to request a change in the number of cycles that the AAX host expects the DSP plug-in to consume.

Note

A plug-in should never apply a DSP algorithm with more demanding resource requirements than what is currently accounted for by the host. To set a higher cycle count value, a plug-in must call [AAX_IController::SetCycleCount\(\)](#), then poll [AAX_IController::GetCycleCount\(\)](#) until the new value has been applied. Once the host has recognized the new cycle count value, the plug-in may apply the more demanding algorithm.

Parameters

in	<i>inWhichCycleCounts</i>	Array of selectors indicating the specific cycle count metrics that should be set. Each selector must be one of: <ul style="list-style-type: none"> • AAX_eProperty_TI_SharedCycleCount • AAX_eProperty_TI_InstanceCycleCount • AAX_eProperty_TI_MaxInstancesPerChip
in	<i>iValues</i>	An array of values requested, one for each of the selected cycle count metrics.
in	<i>numValues</i>	The size of <i>iValues</i>

Todo PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

14.53.2.10 PostPacket()

```
virtual AAX_Result AAX_IACFController::PostPacket (
    AAX_CFieldIndex inFieldIndex,
    const void * inPayloadP,
    uint32_t inPayloadSize ) [pure virtual]
```

CALL: Posts a data packet to the host for routing between plug-in components.

The posted packet is identified with a [AAX_CFieldIndex](#) packet index value, which is equivalent to the target data port's identifier. The packet's payload must have the expected size for the given packet index / data port, as defined when the port is created in [Describe](#). See [AAX_IComponentDescriptor::AddDataInPort\(\)](#).

Warning

Any data structures that will be passed between platforms (for example, sent to a TI DSP in an AAX DSP plug-in) must be properly data-aligned for compatibility across both platforms. See [AAX_ALIGN_FILE_ALG](#) for more information about guaranteeing cross-platform compatibility of data structures used for algorithm processing.

Note

All calls to this method should be made within the scope of [AAX_IEffectParameters::GenerateCoefficients\(\)](#). Calls from outside this method may result in packets not being delivered. See [PT-206161](#)

Parameters

in	<i>inFieldIndex</i>	The packet's destination port
in	<i>inPayloadP</i>	A pointer to the packet's payload data
in	<i>inPayloadSize</i>	The size, in bytes, of the payload data

14.53.2.11 GetCurrentMeterValue()

```
virtual AAX_Result AAX_IACFController::GetCurrentMeterValue (
    AAX_CTypeID inMeterID,
    float * outMeterValue ) const [pure virtual]
```

CALL: Retrieves the current value of a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried
out	<i>outMeterValue</i>	The queried meter's current value

14.53.2.12 GetMeterPeakValue()

```
virtual AAX_Result AAX_IACFController::GetMeterPeakValue (
    AAX_CTypeID inMeterID,
    float * outMeterPeakValue ) const [pure virtual]
```

CALL: Retrieves the currently held peak value of a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried
out	<i>outMeterPeakValue</i>	The queried meter's currently held peak value

14.53.2.13 ClearMeterPeakValue()

```
virtual AAX_Result AAX_IACFController::ClearMeterPeakValue (
    AAX_CTypeID inMeterID ) const [pure virtual]
```

CALL: Clears the peak value from a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being cleared
----	------------------	---------------------------------------

14.53.2.14 GetMeterClipped()

```
virtual AAX_Result AAX_IACFController::GetMeterClipped (
    AAX_CTypeID inMeterID,
    AAX_CBoolean * outClipped ) const [pure virtual]
```

CALL: Retrieves the clipped flag from a host-managed plug-in meter.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried.
out	<i>outClipped</i>	The queried meter's clipped flag.

14.53.2.15 ClearMeterClipped()

```
virtual AAX_Result AAX_IACFController::ClearMeterClipped (
    AAX_CTypeID inMeterID ) const [pure virtual]
```

CALL: Clears the clipped flag from a host-managed plug-in meter.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

in	<i>inMeterID</i>	ID of the meter that is being cleared.
----	------------------	--

14.53.2.16 GetMeterCount()

```
virtual AAX_Result AAX_IACFController::GetMeterCount (
    uint32_t * outMeterCount ) const [pure virtual]
```

CALL: Retrieves the number of host-managed meters registered by a plug-in.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

out	<i>outMeterCount</i>	The number of registered plug-in meters.
-----	----------------------	--

14.53.2.17 GetNextMIDIpacket()

```
virtual AAX_Result AAX_IACFController::GetNextMIDIpacket (
    AAX_CFieldIndex * outPort,
    AAX_CMidiPacket * outPacket ) [pure virtual]
```

CALL: Retrieves MIDI packets for described MIDI nodes.

Parameters

out	<i>outPort</i>	port ID of the MIDI node that has unhandled packet
out	<i>outPacket</i>	The MIDI packet

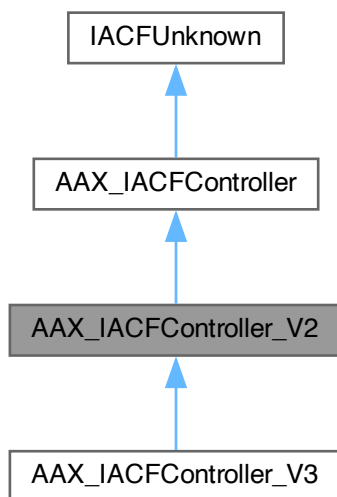
The documentation for this class was generated from the following file:

- [AAX_IACFController.h](#)

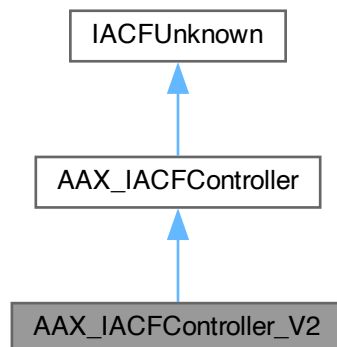
14.54 AAX_IACFController_V2 Class Reference

```
#include <AAX_IACFController.h>
```

Inheritance diagram for AAX_IACFController_V2:



Collaboration diagram for AAX_IACFController_V2:



14.54.1 Description

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.

Public Member Functions

- virtual [AAX_Result SendNotification](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
CALL: Dispatch a notification.
- virtual [AAX_Result GetHybridSignalLatency](#) (int32_t *outSamples) const =0
CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.
- virtual [AAX_Result GetCurrentAutomationTimestamp](#) ([AAX_CTransportCounter](#) *outTimestamp) const =0
CALL: Returns the current automation timestamp if called during the [GenerateCoefficients\(\)](#) call AND the generation of coefficients is being triggered by an automation point instead of immediate changes.
- virtual [AAX_Result GetHostName](#) ([AAX_IString](#) *outHostNameString) const =0
CALL: Returns name of the host application this plug-in instance is being loaded by. This string also typically includes version information.

Public Member Functions inherited from [AAX_IACFController](#)

- virtual [AAX_Result GetEffectID](#) ([AAX_IString](#) *outEffectID) const =0
- virtual [AAX_Result GetSampleRate](#) ([AAX_CSampleRate](#) *outSampleRate) const =0
CALL: Returns the current literal sample rate.
- virtual [AAX_Result GetInputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's input stem format.
- virtual [AAX_Result GetOutputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's output stem format.
- virtual [AAX_Result GetSignalLatency](#) (int32_t *outSamples) const =0

- CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.*

 - virtual [AAX_Result GetCycleCount](#) ([AAX_EProperty](#) inWhichCycleCount, [AAX_CPropertyValue](#) *outNumCycles) const =0

CALL: returns the plug-in's current real-time DSP cycle count.
- virtual [AAX_Result GetTODLocation](#) ([AAX_CTimeOfDay](#) *outTODLocation) const =0

CALL: Returns the current Time Of Day (TOD) of the system.
- virtual [AAX_Result SetSignalLatency](#) (int32_t inNumSamples)=0

CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.
- virtual [AAX_Result SetCycleCount](#) ([AAX_EProperty](#) *inWhichCycleCounts, [AAX_CPropertyValue](#) *iValues, int32_t numValues)=0

CALL: Indicates a change in the plug-in's real-time DSP cycle count.
- virtual [AAX_Result PostPacket](#) ([AAX_CFieldIndex](#) inFieldIndex, const void *inPayloadP, uint32_t inPayloadSize)=0

CALL: Posts a data packet to the host for routing between plug-in components.
- virtual [AAX_Result GetCurrentMeterValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterValue) const =0

CALL: Retrieves the current value of a host-managed plug-in meter.
- virtual [AAX_Result GetMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterPeakValue) const =0

CALL: Retrieves the currently held peak value of a host-managed plug-in meter.
- virtual [AAX_Result ClearMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID) const =0

CALL: Clears the peak value from a host-managed plug-in meter.
- virtual [AAX_Result GetMeterClipped](#) ([AAX_CTypeID](#) inMeterID, [AAX_CBoolean](#) *outClipped) const =0

CALL: Retrieves the clipped flag from a host-managed plug-in meter.
- virtual [AAX_Result ClearMeterClipped](#) ([AAX_CTypeID](#) inMeterID) const =0

CALL: Clears the clipped flag from a host-managed plug-in meter.
- virtual [AAX_Result GetMeterCount](#) (uint32_t *outMeterCount) const =0

CALL: Retrieves the number of host-managed meters registered by a plug-in.
- virtual [AAX_Result GetNextMIDIPacket](#) ([AAX_CFieldIndex](#) *outPort, [AAX_CMidiPacket](#) *outPacket)=0

CALL: Retrieves MIDI packets for described MIDI nodes.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0

Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0

Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0

Decrements reference count.

14.54.2 Member Function Documentation

14.54.2.1 SendNotification()

```
virtual AAX_Result AAX_IACFController_V2::SendNotification (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [pure virtual]
```

CALL: Dispatch a notification.

The notification is handled by the host and may be delivered back to other plug-in components such as the GUI or data model (via [AAX_IEffectGUI::NotificationReceived\(\)](#) or [AAX_IEffectParameters::NotificationReceived\(\)](#), respectively) depending on the notification type.

The host may choose to dispatch the posted notification either synchronously or asynchronously.

See the [AAX_ENotificationEvent](#) documentation for more information.

This method is supported by AAX V2 Hosts only. Check the return code on the return of this function. If the error is [AAX_ERROR_UNIMPLEMENTED](#), your plug-in is being loaded into a host that doesn't support this feature.

Parameters

in	<i>inNotificationType</i>	Type of notification to send
in	<i>inNotificationData</i>	Block of notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

14.54.2.2 GetHybridSignalLatency()

```
virtual AAX_Result AAX_IACFController_V2::GetHybridSignalLatency (
    int32_t * outSamples ) const [pure virtual]
```

CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.

This method provides the number of samples that the AAX host expects the plug-in to delay a signal. The host will use this value when accounting for latency across the system.

Note

This value will generally scale up with sample rate, although it's not a simple multiple due to some fixed overhead. This value will be fixed for any given sample rate regardless of other buffer size settings in the host app.

Parameters

out	<i>outSamples</i>	The number of samples of hybrid signal delay
-----	-------------------	--

14.54.2.3 GetCurrentAutomationTimestamp()

```
virtual AAX_Result AAX_IACFController_V2::GetCurrentAutomationTimestamp (
    AAX_CTransportCounter * outTimestamp ) const [pure virtual]
```

CALL: Returns the current automation timestamp if called during the [GenerateCoefficients\(\)](#) call AND the generation of coefficients is being triggered by an automation point instead of immediate changes.

Note

This function will return 0 if called from outside of [GenerateCoefficients\(\)](#) or if the [GenerateCoefficients\(\)](#) call was initiated due to a non-automated change. In those cases, you can get your sample offset from the transport start using [GetTODLocation\(\)](#).

Parameters

out	<i>outTimestamp</i>	The current coefficient timestamp. Sample count from transport start.
-----	---------------------	---

14.54.2.4 GetHostName()

```
virtual AAX_Result AAX_IACFController_V2::GetHostName (
    AAX_IString * outHostNameString ) const [pure virtual]
```

CALL: Returns name of the host application this plug-in instance is being loaded by. This string also typically includes version information.

Host Compatibility Notes Pro Tools versions from Pro Tools 11.0 to Pro Tools 12.3.1 will return a generic version string to this call. This issue is resolved beginning in Pro Tools 12.4.

Parameters

out	<i>outHostNameString</i>	The name of the current host application.
-----	--------------------------	---

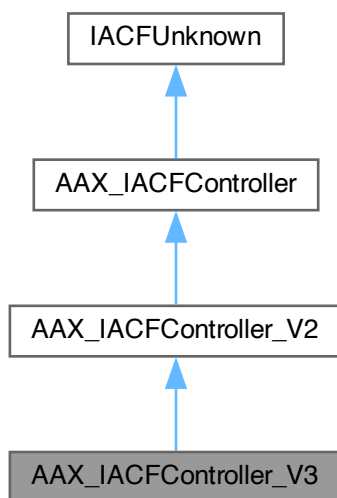
The documentation for this class was generated from the following file:

- [AAX_IACFController.h](#)

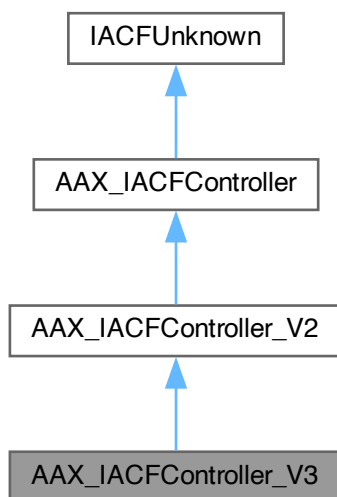
14.55 AAX_IACFController_V3 Class Reference

```
#include <AAX_IACFController.h>
```

Inheritance diagram for AAX_IACFController_V3:



Collaboration diagram for AAX_IACFController_V3:



14.55.1 Description

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.

Public Member Functions

- virtual [AAX_Result](#) [GetPlugInTargetPlatform](#) ([AAX_CTargetPlatform](#) *outTargetPlatform) const =0
CALL: Returns execution platform type, native or TI.
- virtual [AAX_Result](#) [GetIsAudioSuite](#) ([AAX_CBoolean](#) *outIsAudioSuite) const =0
CALL: Returns true for AudioSuite instances.

Public Member Functions inherited from [AAX_IACFController_V2](#)

- virtual [AAX_Result](#) [SendNotification](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
CALL: Dispatch a notification.
- virtual [AAX_Result](#) [GetHybridSignalLatency](#) (int32_t *outSamples) const =0
CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.
- virtual [AAX_Result](#) [GetCurrentAutomationTimestamp](#) ([AAX_CTransportCounter](#) *outTimestamp) const =0
CALL: Returns the current automation timestamp if called during the [GenerateCoefficients\(\)](#) call AND the generation of coefficients is being triggered by an automation point instead of immediate changes.
- virtual [AAX_Result](#) [GetHostName](#) ([AAX_IString](#) *outHostNameString) const =0
CALL: Returns name of the host application this plug-in instance is being loaded by. This string also typically includes version information.

Public Member Functions inherited from [AAX_IACFController](#)

- virtual [AAX_Result](#) [GetEffectID](#) ([AAX_IString](#) *outEffectID) const =0
- virtual [AAX_Result](#) [GetSampleRate](#) ([AAX_CSampleRate](#) *outSampleRate) const =0
CALL: Returns the current literal sample rate.
- virtual [AAX_Result](#) [GetInputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's input stem format.
- virtual [AAX_Result](#) [GetOutputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's output stem format.
- virtual [AAX_Result](#) [GetSignalLatency](#) (int32_t *outSamples) const =0
CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.
- virtual [AAX_Result](#) [GetCycleCount](#) ([AAX_EProperty](#) inWhichCycleCount, [AAX_CPropertyValue](#) *outNumCycles) const =0
CALL: returns the plug-in's current real-time DSP cycle count.
- virtual [AAX_Result](#) [GetTODLocation](#) ([AAX_CTimeOfDay](#) *outTODLocation) const =0
CALL: Returns the current Time Of Day (TOD) of the system.
- virtual [AAX_Result](#) [SetSignalLatency](#) (int32_t inNumSamples)=0
CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.
- virtual [AAX_Result](#) [SetCycleCount](#) ([AAX_EProperty](#) *inWhichCycleCounts, [AAX_CPropertyValue](#) *iValues, int32_t numValues)=0
CALL: Indicates a change in the plug-in's real-time DSP cycle count.
- virtual [AAX_Result](#) [PostPacket](#) ([AAX_CFieldIndex](#) inFieldIndex, const void *inPayloadP, uint32_t inPayloadSize)=0
CALL: Posts a data packet to the host for routing between plug-in components.
- virtual [AAX_Result](#) [GetCurrentMeterValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterValue) const =0
CALL: Retrieves the current value of a host-managed plug-in meter.
- virtual [AAX_Result](#) [GetMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterPeakValue) const =0
CALL: Retrieves the currently held peak value of a host-managed plug-in meter.

- virtual [AAX_Result ClearMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID) const =0
CALL: Clears the peak value from a host-managed plug-in meter.
- virtual [AAX_Result GetMeterClipped](#) ([AAX_CTypeID](#) inMeterID, [AAX_CBoolean](#) *outClipped) const =0
CALL: Retrieves the clipped flag from a host-managed plug-in meter.
- virtual [AAX_Result ClearMeterClipped](#) ([AAX_CTypeID](#) inMeterID) const =0
CALL: Clears the clipped flag from a host-managed plug-in meter.
- virtual [AAX_Result GetMeterCount](#) (uint32_t *outMeterCount) const =0
CALL: Retrieves the number of host-managed meters registered by a plug-in.
- virtual [AAX_Result GetNextMIDIPacket](#) ([AAX_CFieldIndex](#) *outPort, [AAX_CMidiPacket](#) *outPacket)=0
CALL: Retrieves MIDI packets for described MIDI nodes.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.55.2 Member Function Documentation

14.55.2.1 GetPlugInTargetPlatform()

```
virtual AAX\_Result AAX_IACFController_V3::GetPlugInTargetPlatform (
    AAX\_CTargetPlatform * outTargetPlatform ) const [pure virtual]
```

CALL: Returns execution platform type, native or TI.

Parameters

out	<i>outTargetPlatform</i>	The type of the current execution platform as one of AAX_ETargetPlatform .
-----	--------------------------	--

14.55.2.2 GetIsAudioSuite()

```
virtual AAX\_Result AAX_IACFController_V3::GetIsAudioSuite (
    AAX\_CBoolean * outIsAudioSuite ) const [pure virtual]
```

CALL: Returns true for AudioSuite instances.

Parameters

out	<i>outIsAudioSuite</i>	The boolean flag which indicate true for AudioSuite instances.
-----	------------------------	--

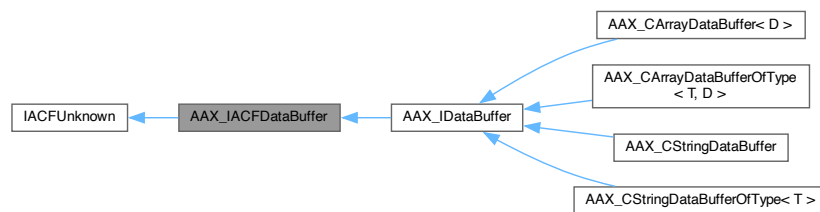
The documentation for this class was generated from the following file:

- [AAX_IACFController.h](#)

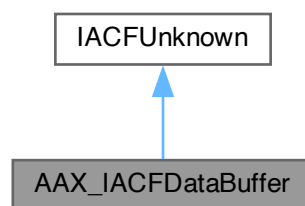
14.56 AAX_IACFDataBuffer Class Reference

```
#include <AAX_IACFDataBuffer.h>
```

Inheritance diagram for AAX_IACFDataBuffer:



Collaboration diagram for AAX_IACFDataBuffer:



14.56.1 Description

Versioned interface for reference counted data buffers.

This interface is intended to be used for passing arbitrary blocks of data across the binary boundary and allowing the receiver to take ownership of the allocated memory.

Public Member Functions

- virtual [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result Size](#) (int32_t *oSize) const =0
- virtual [AAX_Result Data](#) (void const **oBuffer) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.56.2 Member Function Documentation

14.56.2.1 Type()

```
virtual AAX\_Result AAX_IACFDataBuffer::Type (
    AAX\_CTypeID * oType ) const [pure virtual]
```

The type of data contained in this buffer

This identifier must be sufficient for a client that knows the type to correctly interpret and use the data.

Implemented in [AAX_CArrayDataBufferOfType< T, D >](#), [AAX_CArrayDataBuffer< D >](#), [AAX_CStringDataBufferOfType< T >](#), and [AAX_CStringDataBuffer](#).

14.56.2.2 Size()

```
virtual AAX\_Result AAX_IACFDataBuffer::Size (
    int32_t * oSize ) const [pure virtual]
```

The number of bytes of data in this buffer

Implemented in [AAX_CArrayDataBufferOfType< T, D >](#), [AAX_CArrayDataBuffer< D >](#), [AAX_CStringDataBufferOfType< T >](#), and [AAX_CStringDataBuffer](#).

14.56.2.3 Data()

```
virtual AAX_Result AAX_IACFDataBuffer::Data (
    void const ** oBuffer ) const [pure virtual]
```

The buffer of data

Implemented in [AAX_CArrayDataBufferOfType< T, D >](#), [AAX_CArrayDataBuffer< D >](#), [AAX_CStringDataBufferOfType< T >](#), and [AAX_CStringDataBuffer](#).

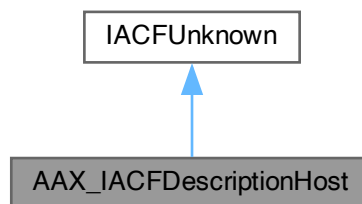
The documentation for this class was generated from the following file:

- [AAX_IACFDataBuffer.h](#)

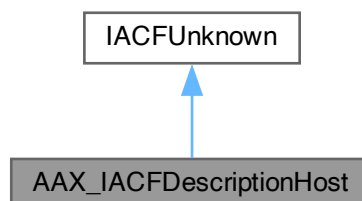
14.57 AAX_IACFDescriptionHost Class Reference

```
#include <AAX_IACFDescriptionHost.h>
```

Inheritance diagram for AAX_IACFDescriptionHost:



Collaboration diagram for AAX_IACFDescriptionHost:



14.57.1 Description

Interface to host services provided during plug-in description

Public Member Functions

- virtual [AAX_Result](#) [AcquireFeatureProperties](#) (const [AAX_Feature_UID](#) &inFeatureID, [IACFUnknown](#) **outFeatureProperties)=0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.57.2 Member Function Documentation

14.57.2.1 [AcquireFeatureProperties\(\)](#)

```
virtual AAX\_Result AAX_IACFDescriptionHost::AcquireFeatureProperties (
    const AAX\_Feature\_UID & inFeatureID,
    IACFUnknown ** outFeatureProperties ) [pure virtual]
```

outFeatureProperties must support [AAX_IACFFeatureInfo](#) const methods

See also

[AAX_IDescriptionHost::AcquireFeatureProperties\(\)](#)

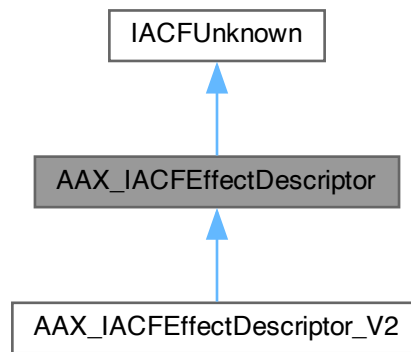
The documentation for this class was generated from the following file:

- [AAX_IACFDescriptionHost.h](#)

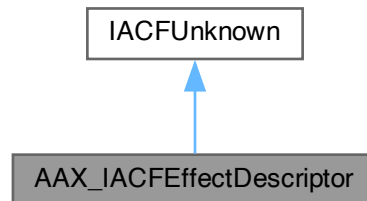
14.58 AAX_IACFEffectorDescriptor Class Reference

```
#include <AAX_IACFEffectorDescriptor.h>
```

Inheritance diagram for AAX_IACFEffectorDescriptor:



Collaboration diagram for AAX_IACFEffectorDescriptor:



14.58.1 Description

Versioned interface for an [AAX_IEffectDescriptor](#).

Public Member Functions

- virtual [AAX_Result](#) [AddComponent](#) ([IACFUnknown](#) *inComponentDescriptor)=0
Add a component to an instance of a component descriptor.
- virtual [AAX_Result](#) [AddName](#) (const char *inPlugInName)=0
Add a name to the Effect.

- virtual [AAX_Result AddCategory](#) (uint32_t inCategory)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result AddCategoryBypassParameter](#) (uint32_t inCategory, [AAX_CParamID](#) inParamID)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result AddProcPtr](#) (void *inProcPtr, [AAX_CProcPtrID](#) inProcID)=0
Add a process pointer.
- virtual [AAX_Result SetProperties](#) ([IACFUnknown](#) *inProperties)=0
Set the properties of a new property map.
- virtual [AAX_Result AddResourceInfo](#) ([AAX_EResourceType](#) inResourceType, const char *inInfo)=0
Set resource file info.
- virtual [AAX_Result AddMeterDescription](#) ([AAX_CTypeID](#) inMeterID, const char *inMeterName, [IACFUnknown](#) *inProperties)=0
Add name and property map to meter with given ID.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.58.2 Member Function Documentation

14.58.2.1 AddComponent()

```
virtual AAX\_Result AAX_IACFEffectorDescriptor::AddComponent (
    IACFUnknown * inComponentDescriptor ) [pure virtual]
```

Add a component to an instance of a component descriptor.

Unlike with [AAX_ICollection::AddEffect\(\)](#), the [AAX_IEffectorDescriptor](#) does not take ownership of the [AAX_IComponentDescriptor](#) that is passed to it in this method. The host copies out the contents of this descriptor, and thus the plug-in may re-use the same descriptor object when creating additional similar components.

Parameters

in	<i>inComponentDescriptor</i>	
----	------------------------------	--

14.58.2.2 AddName()

```
virtual AAX\_Result AAX_IACFEffectorDescriptor::AddName (
    const char * inPlugInName ) [pure virtual]
```

Add a name to the Effect.

May be called multiple times to add abbreviated Effect names.

Note

Every Effect must include at least one name variant with 31 or fewer characters, plus a null terminating character

Parameters

in	<i>inPlugInName</i>	The name assigned to the plug-in.
----	---------------------	-----------------------------------

14.58.2.3 AddCategory()

```
virtual AAX_Result AAX_IACEffectDescriptor::AddCategory (
    uint32_t inCategory ) [pure virtual]
```

Add a category to your plug-in. See [AAX_EPlugInCategory](#).

Parameters

in	<i>inCategory</i>	One of the categories for the plug-in.
----	-------------------	--

14.58.2.4 AddCategoryBypassParameter()

```
virtual AAX_Result AAX_IACEffectDescriptor::AddCategoryBypassParameter (
    uint32_t inCategory,
    AAX_CParamID inParamID ) [pure virtual]
```

Add a category to your plug-in. See [AAX_EPlugInCategory](#).

Parameters

in	<i>inCategory</i>	One of the categories for the plug-in.
in	<i>inParamID</i>	The parameter ID of the parameter used to bypass the category section of the plug-in.

14.58.2.5 AddProcPtr()

```
virtual AAX_Result AAX_IACEffectDescriptor::AddProcPtr (
    void * inProcPtr,
    AAX_CProcPtrID inProcID ) [pure virtual]
```

Add a process pointer.

Parameters

in	<i>inProcPtr</i>	A process pointer.
in	<i>inProcID</i>	A process ID.

14.58.2.6 SetProperty()

```
virtual AAX_Result AAX_IACEffectDescriptor::SetProperties (
    IACFUnknown * inProperties ) [pure virtual]
```

Set the properties of a new property map.

Parameters

in	<i>inProperties</i>	Description
----	---------------------	-------------

14.58.2.7 AddResourceInfo()

```
virtual AAX_Result AAX_IACEffectDescriptor::AddResourceInfo (
    AAX_EResourceType inResourceType,
    const char * inInfo ) [pure virtual]
```

Set resource file info.

Parameters

in	<i>inResourceType</i>	See AAX_EResourceType.
in	<i>inInfo</i>	Definition varies on the resource type.

14.58.2.8 AddMeterDescription()

```
virtual AAX_Result AAX_IACEffectDescriptor::AddMeterDescription (
    AAX_CTypeID inMeterID,
    const char * inMeterName,
    IACFUnknown * inProperties ) [pure virtual]
```

Add name and property map to meter with given ID.

Parameters

in	<i>inMeterID</i>	The ID of the meter being described.
in	<i>inMeterName</i>	The name of the meter.
in	<i>inProperties</i>	The property map containing meter related data such as meter type, orientation, etc.

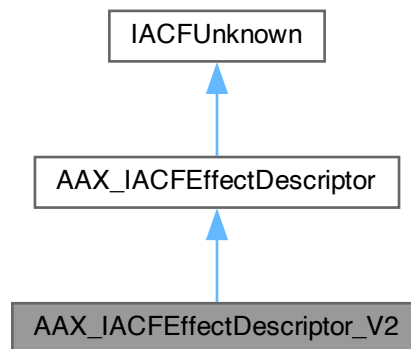
The documentation for this class was generated from the following file:

- [AAX_IACFEffectorDescriptor.h](#)

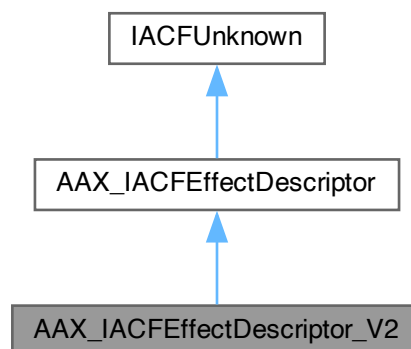
14.59 AAX_IACFEffectorDescriptor_V2 Class Reference

```
#include <AAX_IACFEffectorDescriptor.h>
```

Inheritance diagram for AAX_IACFEffectorDescriptor_V2:



Collaboration diagram for AAX_IACFEffectorDescriptor_V2:



14.59.1 Description

Versioned interface for an [AAX_IEffectDescriptor](#).

Public Member Functions

- virtual [AAX_Result AddControlMIDINode](#) ([AAX_CTypeID](#) inNodeID, [AAX_EMIDINodeType](#) inNodeType, const char inNodeName[], uint32_t inChannelMask)=0
Add a control MIDI node to the plug-in data model.

Public Member Functions inherited from [AAX_IACFEffectorDescriptor](#)

- virtual [AAX_Result AddComponent](#) ([IACFUnknown](#) *inComponentDescriptor)=0
Add a component to an instance of a component descriptor.
- virtual [AAX_Result AddName](#) (const char *inPlugInName)=0
Add a name to the Effect.
- virtual [AAX_Result AddCategory](#) (uint32_t inCategory)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result AddCategoryBypassParameter](#) (uint32_t inCategory, [AAX_CParamID](#) inParamID)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result AddProcPtr](#) (void *inProcPtr, [AAX_CProcPtrID](#) inProcID)=0
Add a process pointer.
- virtual [AAX_Result SetProperties](#) ([IACFUnknown](#) *inProperties)=0
Set the properties of a new property map.
- virtual [AAX_Result AddResourceInfo](#) ([AAX_EResourceType](#) inResourceType, const char *inInfo)=0
Set resource file info.
- virtual [AAX_Result AddMeterDescription](#) ([AAX_CTypeID](#) inMeterID, const char *inMeterName, [IACFUnknown](#) *inProperties)=0
Add name and property map to meter with given ID.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.59.2 Member Function Documentation

14.59.2.1 AddControlMIDINode()

```
virtual AAX\_Result AAX_IACFEffectorDescriptor_V2::AddControlMIDINode (
    AAX\_CTypeID inNodeID,
    AAX\_EMIDINodeType inNodeType,
    const char inNodeName[],
    uint32_t inChannelMask ) [pure virtual]
```

Add a control MIDI node to the plug-in data model.

- This MIDI node may receive note data as well as control data.
- To send MIDI data to the plug-in's algorithm, use [AAX_IComponentDescriptor::AddMIDINode\(\)](#).

See also

[AAX_IACFEffectorParameters_V2::UpdateControlMIDINodes\(\)](#)

Parameters

in	<i>inNodeID</i>	The ID for the new control MIDI node.
in	<i>inNodeType</i>	The type of the node.
in	<i>inNodeName</i>	The name of the node.
in	<i>inChannelMask</i>	The bit mask for required nodes channels (up to 16) or required global events for global node.

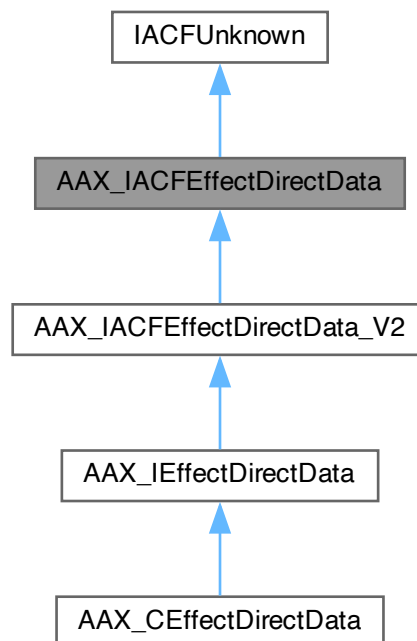
The documentation for this class was generated from the following file:

- [AAX_IACFEffectorDescriptor.h](#)

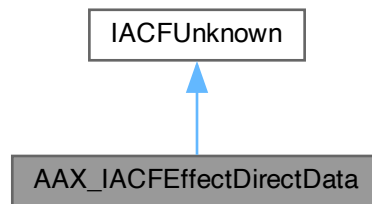
14.60 AAX_IACFEffectorDirectData Class Reference

```
#include <AAX_IACFEffectorDirectData.h>
```

Inheritance diagram for AAX_IACFEffectorDirectData:



Collaboration diagram for AAX_IACFEfffectDirectData:



14.60.1 Description

Optional interface for direct access to a plug-in's alg memory.

Direct data access allows a plug-in to directly manipulate the data in its algorithm's private data blocks. The callback methods in this interface provide a safe context from which this kind of access may be attempted.

Public Member Functions

Initialization and uninitialization

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Main initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Main uninitialization.

Safe data update callbacks

These callbacks provide a safe context from which to directly access the algorithm's private data blocks. Each callback is called regularly with roughly the schedule of its corresponding [AAX_IEffectParameters](#) counterpart.

Note

Do not attempt to directly access the algorithm's data from outside these callbacks. Instead, use the packet system to route data to the algorithm using the AAX host's buffered data transfer facilities.

- virtual [AAX_Result TimerWakeup](#) ([IACFUnknown](#) *iDataAccessInterface)=0
Periodic wakeup callback for idle-time operations.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.60.2 Member Function Documentation

14.60.2.1 Initialize()

```
virtual AAX_Result AAX_IACFEffEffectDirectData::Initialize (
    IACFUnknown * iController ) [pure virtual]
```

Main initialization.

Called when the interface is created

Parameters

in	<i>iController</i>	A versioned reference that resolves to an AAX_IController interface
----	--------------------	---

Implemented in [AAX_CEffectDirectData](#).

14.60.2.2 Uninitialize()

```
virtual AAX_Result AAX_IACFEffEffectDirectData::Uninitialize ( ) [pure virtual]
```

Main uninitialization.

Called when the interface is destroyed.

Implemented in [AAX_CEffectDirectData](#).

14.60.2.3 TimerWakeup()

```
virtual AAX_Result AAX_IACFEffEffectDirectData::TimerWakeup (
    IACFUnknown * iDataAccessInterface ) [pure virtual]
```

Periodic wakeup callback for idle-time operations.

Direct alg data updates must be triggered from this method.

This method is called from the host using a non-main thread. In general, it should be driven at approximately one call per 30 ms. However, the wakeup is not guaranteed to be called at any regular interval - for example, it could be held off by a high real-time processing load - and there is no host contract regarding maximum latency between wakeup calls.

This wakeup thread runs continuously and cannot be armed/disarmed or by the plug-in.

Parameters

in	<i>iDataAccessInterface</i>	Reference to the direct access interface.
----	-----------------------------	---

Note

It is not safe to save this address or call the methods in this interface from other threads.

Implemented in [AAX_CEffectDirectData](#).

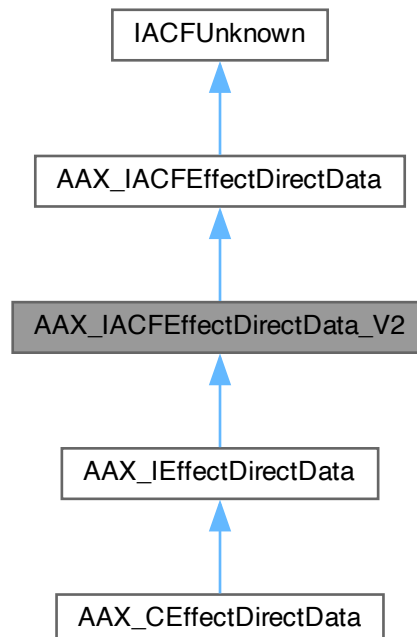
The documentation for this class was generated from the following file:

- [AAX_IACFEfffectDirectData.h](#)

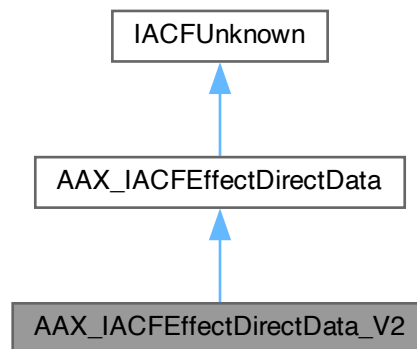
14.61 AAX_IACFEfffectDirectData_V2 Class Reference

```
#include <AAX_IACFEfffectDirectData.h>
```

Inheritance diagram for AAX_IACFEfffectDirectData_V2:



Collaboration diagram for AAX_IACFEfffectDirectData_V2:



Public Member Functions

AAX host and plug-in event notification

- virtual [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.

Public Member Functions inherited from [AAX_IACFEfffectDirectData](#)

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Main initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Main uninitialization.
- virtual [AAX_Result TimerWakeup](#) ([IACFUnknown](#) *iDataAccessInterface)=0
Periodic wakeup callback for idle-time operations.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.61.1 Member Function Documentation

14.61.1.1 NotificationReceived()

```
virtual AAX_Result AAX_IACFEffEffectDirectData_V2::NotificationReceived (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [pure virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- some notifications are sent only to the plug-in GUI while other notifications are sent only to the plug-in data model. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

A plug-in may also dispatch custom notifications using [AAX_IController::SendNotification\(\)](#). Custom notifications will be posted back to the plug-in's other objects which support a [NotificationReceived\(\)](#) method (e.g. the GUI).

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implemented in [AAX_CEffectDirectData](#).

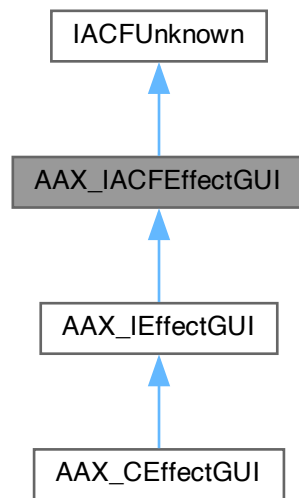
The documentation for this class was generated from the following file:

- [AAX_IACFEffEffectDirectData.h](#)

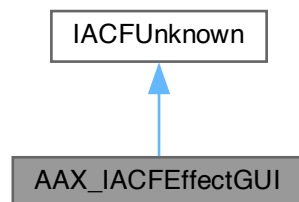
14.62 AAX_IACFEffEffectGUI Class Reference

```
#include <AAX_IACFEffEffectGUI.h>
```

Inheritance diagram for AAX_IACFEEffectGUI:



Collaboration diagram for AAX_IACFEEffectGUI:



14.62.1 Description

The interface for a AAX Plug-in's GUI (graphical user interface).

This is the interface for an instance of a plug-in's GUI that gets exposed to the host application. The AAX host interacts with your plug-in's GUI via this interface. See [GUI interface](#).

The plug-in's implementation of this interface is responsible for managing the plug-in's window and graphics objects and for defining the interactions between GUI views and the plug-in's data model.

At [initialization](#), the host provides this interface with a reference to [AAX_IController](#). The GUI may use this controller to retrieve a pointer to the plug-in's [AAX_IEffectParameters](#) interface, allowing the GUI to request changes to the

plug-in's data model in response to view events. In addition, the controller provides a means of querying information from the host such as stem format or sample rate

When managing a plug-in's GUI it is important to remember that this is just one of many possible sets of views for the plug-in's parameters. Other views and editors, such as automation lanes or control surfaces, also have the ability to synchronously interact with the plug-in's abstract data model interface. Therefore, the GUI should not take asymmetric control over the data model, act as a secondary data model, or otherwise assume exclusive ownership of the plug-in's state. In general, the data model's abstraction to a pure virtual interface will protect against such aberrations, but this remains an important consideration when managing sophisticated GUI interactions.

You will most likely inherit your implementation of this interface from [AAX_CEffectGUI](#), a default implementation that provides basic GUI functionality and which you can override and customize as needed.

The SDK includes several examples of how the GUI interface may be extended and implemented in order to provide support for third-party frameworks. These examples can be found in the /Extensions/GUI directory in the SDK.

Note

Your implementation of this interface must inherit from [AAX_IEffectGUI](#).

Legacy Porting Notes In the legacy plug-in SDK, these methods were found in CProcess and CEffectProcess. For additional CProcess methods, see [AAX_IEffectParameters](#).

Public Member Functions

Initialization and uninitialization

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Main GUI initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Main GUI uninitialization.

AAX host and plug-in event notification

- virtual [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.

View accessors

- virtual [AAX_Result SetViewContainer](#) ([IACFUnknown](#) *iViewContainer)=0
Provides a handle to the main plug-in window.
- virtual [AAX_Result GetViewSize](#) ([AAX_Point](#) *oViewSize) const =0
Retrieves the size of the plug-in window.

GUI update methods

- virtual [AAX_Result Draw](#) ([AAX_Rect](#) *iDrawRect)=0
DEPRECATED, Not called from host any longer. Your chosen graphics framework should be directly handling draw events from the OS.
- virtual [AAX_Result TimerWakeup](#) ()=0
Periodic wakeup callback for idle-time operations.
- virtual [AAX_Result ParameterUpdated](#) ([AAX_CParamID](#) inParamID)=0
Notifies the GUI that a parameter value has changed.

Host interface methods

Miscellaneous methods to provide host-specific functionality

- virtual [AAX_Result GetCustomLabel](#) ([AAX_EPlugInStrings](#) iSelector, [AAX_IString](#) *oString) const =0
Called by host application to retrieve a custom plug-in string.
- virtual [AAX_Result SetControlHighlightInfo](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iIsHighlighted, [AAX_EHighlightColor](#) iColor)=0
Called by host application. Indicates that a control widget should be updated with a highlight color.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.62.2 Member Function Documentation

14.62.2.1 Initialize()

```
virtual AAX\_Result AAX_IACFEEffectGUI::Initialize (
    IACFUnknown * iController ) [pure virtual]
```

Main GUI initialization.

Called when the GUI is created

Parameters

in	<i>iController</i>	A versioned reference that resolves to an AAX_IController interface
----	--------------------	---

Implemented in [AAX_CEffectGUI](#).

14.62.2.2 Uninitialize()

```
virtual AAX\_Result AAX_IACFEEffectGUI::Uninitialize ( ) [pure virtual]
```

Main GUI uninitialization.

Called when the GUI is destroyed. Frees the GUI.

Implemented in [AAX_CEffectGUI](#).

14.62.2.3 NotificationReceived()

```
virtual AAX_Result AAX_IACEffectGUI::NotificationReceived (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [pure virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- some notifications are sent only to the plug-in GUI while other notifications are sent only to the plug-in data model. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

A plug-in may also dispatch custom notifications using [AAX_IController::SendNotification\(\)](#). Custom notifications will be posted back to the plug-in's other objects which support a [NotificationReceived\(\)](#) method (e.g. the data model).

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implemented in [AAX_CEffectGUI](#).

14.62.2.4 SetViewContainer()

```
virtual AAX_Result AAX_IACEffectGUI::SetViewContainer (
    IACFUnknown * iViewContainer ) [pure virtual]
```

Provides a handle to the main plug-in window.

Parameters

in	<i>iViewContainer</i>	An AAX_IViewContainer providing a native handle to the plug-in's window
----	-----------------------	---

Implemented in [AAX_CEffectGUI](#).

14.62.2.5 GetViewSize()

```
virtual AAX_Result AAX_IACFEEffectGUI::GetViewSize (
    AAX_Point * oViewSize ) const [pure virtual]
```

Retrieves the size of the plug-in window.

See also

[AAX_IViewContainer::SetViewSize\(\)](#)

Parameters

out	<i>oViewSize</i>	The size of the plug-in window as a point (width, height)
-----	------------------	---

Implemented in [AAX_CEffectGUI](#).

14.62.2.6 Draw()

```
virtual AAX_Result AAX_IACFEEffectGUI::Draw (
    AAX_Rect * iDrawRect ) [pure virtual]
```

DEPRECATED, Not called from host any longer. Your chosen graphics framework should be directly handling draw events from the OS.

Implemented in [AAX_CEffectGUI](#).

14.62.2.7 TimerWakeup()

```
virtual AAX_Result AAX_IACFEEffectGUI::TimerWakeup ( ) [pure virtual]
```

Periodic wakeup callback for idle-time operations.

GUI animation events such as meter updates should be triggered from this method.

This method is called from the host's main thread. In general, it should be driven at approximately one call per 30 ms. However, the wakeup is not guaranteed to be called at any regular interval - for example, it could be held off by a high real-time processing load - and there is no host contract regarding maximum latency between wakeup calls.

This wakeup runs continuously and cannot be armed/disarmed by the plug-in.

Implemented in [AAX_CEffectGUI](#).

14.62.2.8 ParameterUpdated()

```
virtual AAX_Result AAX_IACFEEffectGUI::ParameterUpdated (
    AAX_CParamID inParamID ) [pure virtual]
```

Notifies the GUI that a parameter value has changed.

This method is called by the host whenever a parameter value has been modified

This method may be called on a non-main thread

Implemented in [AAX_CEffectGUI](#).

14.62.2.9 GetCustomLabel()

```
virtual AAX_Result AAX_IACFEEffectGUI::GetCustomLabel (
    AAX_EPlugInStrings iSelector,
    AAX_IString * oString ) const [pure virtual]
```

Called by host application to retrieve a custom plug-in string.

If no string is provided then the host's default will be used.

Parameters

in	<i>iSelector</i>	The requested strong. One of AAX_EPlugInStrings
out	<i>oString</i>	The plug-in's custom value for the requested string

Implemented in [AAX_CEffectGUI](#).

14.62.2.10 SetControlHighlightInfo()

```
virtual AAX_Result AAX_IACFEEffectGUI::SetControlHighlightInfo (
    AAX_CParamID iParameterID,
    AAX_CBoolean iIsHighlighted,
    AAX_EHighlightColor iColor ) [pure virtual]
```

Called by host application. Indicates that a control widget should be updated with a highlight color.

Todo Document this method

Legacy Porting Notes This method was re-named from `SetControlHighliteInfo()`, its name in the legacy plug-in SDK.

Parameters

in	<i>iParameterID</i>	ID of parameter whose widget(s) must be highlighted
in	<i>ilIsHighlighted</i>	True if turning highlight on, false if turning it off
in	<i>iColor</i>	Desired highlight color. One of AAX_EHighlightColor

Implemented in [AAX_CEffectGUI](#).

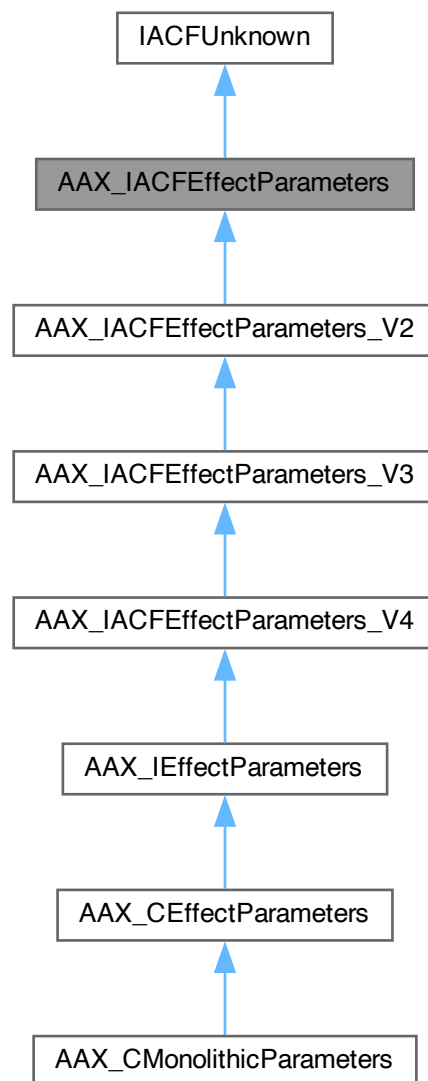
The documentation for this class was generated from the following file:

- [AAX_IACFEEffectGUI.h](#)

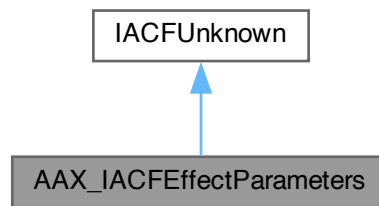
14.63 AAX_IACFEEffectParameters Class Reference

```
#include <AAX_IACFEEffectParameters.h>
```


Inheritance diagram for AAX_IACFEffEffectParameters:



Collaboration diagram for AAX_IACFEffEffectParameters:



14.63.1 Description

The interface for an AAX Plug-in's data model.

This is the interface for an instance of a plug-in's data model that gets exposed to the host application. The AAX host interacts with your plug-in's data model via this interface, which includes methods that store and update of your plug-in's internal data. See [Data model interface](#).

Note

Your implementation of this interface must inherit from [AAX_IEffectParameters](#).

Todo Add documentation for expected error state return values

Public Member Functions

Initialization and uninitialization

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Main data model initialization. Called when plug-in instance is first instantiated.
- virtual [AAX_Result Uninitialize](#) ()=0
Main data model uninitialization.

AAX host and plug-in event notification

- virtual [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.

Parameter information

These methods are used by the AAX host to retrieve information about the plug-in's data model.

For information about adding parameters to the plug-in and otherwise modifying the plug-in's data model, see [AAX_CParameterManager](#). For information about parameters, see [AAX_IParameter](#).

- virtual [AAX_Result GetNumberOfParameters](#) (int32_t *oNumControls) const =0
CALL: Retrieves the total number of plug-in parameters.
- virtual [AAX_Result GetMasterBypassParameter](#) (AAX_IString *oIDString) const =0
CALL: Retrieves the ID of the plug-in's Master Bypass parameter.
- virtual [AAX_Result GetParameterIsAutomatable](#) (AAX_CParamID iParameterID, AAX_CBoolean *oAutomatable) const =0
CALL: Retrieves information about a parameter's automatable status.
- virtual [AAX_Result GetParameterNumberOfSteps](#) (AAX_CParamID iParameterID, int32_t *oNumSteps) const =0
CALL: Retrieves the number of discrete steps for a parameter.
- virtual [AAX_Result GetParameterName](#) (AAX_CParamID iParameterID, AAX_IString *oName) const =0
CALL: Retrieves the full name for a parameter.
- virtual [AAX_Result GetParameterNameOfLength](#) (AAX_CParamID iParameterID, AAX_IString *oName, int32_t iNameLength) const =0
CALL: Retrieves an abbreviated name for a parameter.
- virtual [AAX_Result GetParameterDefaultNormalizedValue](#) (AAX_CParamID iParameterID, double *oValue) const =0
CALL: Retrieves default value of a parameter.
- virtual [AAX_Result SetParameterDefaultNormalizedValue](#) (AAX_CParamID iParameterID, double iValue)=0
CALL: Sets the default value of a parameter.
- virtual [AAX_Result GetParameterType](#) (AAX_CParamID iParameterID, AAX_EParameterType *oParameterType) const =0
CALL: Retrieves the type of a parameter.
- virtual [AAX_Result GetParameterOrientation](#) (AAX_CParamID iParameterID, AAX_EParameterOrientation *oParameterOrientation) const =0
CALL: Retrieves the orientation that should be applied to a parameter's controls.
- virtual [AAX_Result GetParameter](#) (AAX_CParamID iParameterID, AAX_IParameter **oParameter)=0
CALL: Retrieves an arbitrary setting within a parameter.
- virtual [AAX_Result GetParameterIndex](#) (AAX_CParamID iParameterID, int32_t *oControllIndex) const =0
CALL: Retrieves the index of a parameter.
- virtual [AAX_Result GetParameterIDFromIndex](#) (int32_t iControllIndex, AAX_IString *oParameterIDString) const =0
CALL: Retrieves the ID of a parameter.
- virtual [AAX_Result GetParameterValueInfo](#) (AAX_CParamID iParameterID, int32_t iSelector, int32_t *oValue) const =0
CALL: Retrieves a property of a parameter.

Parameter setters and getters

These methods are used by the AAX host and by the plug-in's UI to retrieve and modify the values of the plug-in's parameters.

Note

The parameter setters in this section may generate asynchronous requests.

- virtual [AAX_Result GetParameterValueFromString](#) (AAX_CParamID iParameterID, double *oValue, const AAX_IString &iValueString) const =0
CALL: Converts a value string to a value.
- virtual [AAX_Result GetParameterStringFromValue](#) (AAX_CParamID iParameterID, double iValue, AAX_IString *oValueString, int32_t iMaxLength) const =0
CALL: Converts a normalized parameter value into a string representing its corresponding real value.
- virtual [AAX_Result GetParameterValueString](#) (AAX_CParamID iParameterID, AAX_IString *oValueString, int32_t iMaxLength) const =0
CALL: Retrieves the value string associated with a parameter's current value.
- virtual [AAX_Result GetParameterNormalizedValue](#) (AAX_CParamID iParameterID, double *oValuePtr) const =0
CALL: Retrieves a parameter's current value.
- virtual [AAX_Result SetParameterNormalizedValue](#) (AAX_CParamID iParameterID, double iValue)=0

CALL: Sets the specified parameter to a new value.

- virtual [AAX_Result SetParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0

CALL: Sets the specified parameter to a new value relative to its current value.

Automated parameter helpers

These methods are used to lock and unlock automation system 'resources' when updating automatable parameters.

Note

You should never need to override these methods to extend their behavior beyond what is provided in [AAX_CEffectParameters](#) and [AAX_IParameter](#)

- virtual [AAX_Result TouchParameter](#) ([AAX_CParamID](#) iParameterID)=0
"Touches" (locks) a parameter in the automation system to a particular control in preparation for updates
- virtual [AAX_Result ReleaseParameter](#) ([AAX_CParamID](#) iParameterID)=0
Releases a parameter from a "touched" state.
- virtual [AAX_Result UpdateParameterTouch](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iTouch↔State)=0
Sets a "touched" state on a parameter.

Asynchronous parameter update methods

These methods are called by the AAX host when parameter values have been updated. They are called by the host and can be triggered by other plug-in modules via calls to [AAX_IParameter](#)'s *SetValue* methods, e.g. [SetValueWithFloat\(\)](#)

These methods are responsible for updating parameter values.

Do not call these methods directly! To ensure proper synchronization and to avoid problematic dependency chains, other methods (e.g. [SetParameterNormalizedValue\(\)](#)) and components (e.g. [AAX_IEffectGUI](#)) should always call a *SetValue* method on [AAX_IParameter](#) to update parameter values. The *SetValue* method will properly manage automation locks and other system resources.

- virtual [AAX_Result UpdateParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_EUpdateSource](#) iSource)=0
Updates a single parameter's state to its current value.
- virtual [AAX_Result UpdateParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double i↔Value)=0
Updates a single parameter's state to its current value, as a difference with the parameter's previous value.
- virtual [AAX_Result GenerateCoefficients](#) ()=0
Generates and dispatches new coefficient packets.

State reset handlers

- virtual [AAX_Result ResetFieldData](#) ([AAX_CFieldIndex](#) inFieldIndex, void *oData, uint32_t inDataSize) const =0
Called by the host to reset a private data field in the plug-in's algorithm.

Chunk methods

These methods are used to save and restore collections of plug-in state information, known as chunks. Chunks are used by the host when saving or restoring presets and session settings and when providing "compare" functionality for plug-ins.

The default implementation of these methods in [AAX_CEffectParameters](#) supports a single chunk that includes state information for all of the plug-in's registered parameters. Override all of these methods to add support for additional chunks in your plug-in, for example if your plug-in contains any persistent state that is not encapsulated by its set of registered parameters.

Warning

Remember that plug-in chunk data may be loaded on a different platform from the one where it is saved. All data structures in the chunk must be properly data-aligned for compatibility across all platforms that the plug-in supports. See [AAX_ALIGN_FILE_ALG](#) for notes about common cross-platform pitfalls for data structure alignment.

For reference, see also:

- [AAX_CChunkDataParser](#)
- [AAX_SPlugInChunk](#)
- virtual [AAX_Result GetNumberOfChunks](#) (int32_t *oNumChunks) const =0
Retrieves the number of chunks used by this plug-in.
- virtual [AAX_Result GetChunkIDFromIndex](#) (int32_t iIndex, [AAX_CTypeID](#) *oChunkID) const =0
Retrieves the ID associated with a chunk index.
- virtual [AAX_Result GetChunkSize](#) ([AAX_CTypeID](#) iChunkID, uint32_t *oSize) const =0
Get the size of the data structure that can hold all of a chunk's information.
- virtual [AAX_Result GetChunk](#) ([AAX_CTypeID](#) iChunkID, [AAX_SPlugInChunk](#) *oChunk) const =0
Fills a block of data with chunk information representing the plug-in's current state.
- virtual [AAX_Result SetChunk](#) ([AAX_CTypeID](#) iChunkID, const [AAX_SPlugInChunk](#) *iChunk)=0
Restores a set of plug-in parameters based on chunk information.
- virtual [AAX_Result CompareActiveChunk](#) (const [AAX_SPlugInChunk](#) *iChunkP, [AAX_CBoolean](#) *ols↔ Equal) const =0
Determine if a chunk represents settings that are equivalent to the plug-in's current state.
- virtual [AAX_Result GetNumberOfChanges](#) (int32_t *oNumChanges) const =0
Retrieves the number of parameter changes made since the plug-in's creation.

Thread methods

- virtual [AAX_Result TimerWakeup](#) ()=0
Periodic wakeup callback for idle-time operations.

Auxiliary UI methods

- virtual [AAX_Result GetCurveData](#) ([AAX_CTypeID](#) iCurveType, const float *iValues, uint32_t iNumValues, float *oValues) const =0
Generate a set of output values based on a set of given input values.

Custom data methods

These functions exist as a proxiable way to move data between different modules (e.g. [AAX_IEffectParameters](#) and [AAX_IEffectGUI](#).) Using these, the GUI can query any data through [GetCustomData\(\)](#) with a plug-in defined *typeID*, *void** and size. This has an advantage over just sharing memory in that this function can work as a remote proxy as we enable those sorts of features later in the platform. Likewise, the GUI can also set arbitrary data on the data model by using the [SetCustomData\(\)](#) function with the same idea.

Note

These are plug-in internal only. They are not called from the host right now, or likely ever.

- virtual [AAX_Result GetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, void *oData, uint32_t *oDataWritten) const =0
An optional interface hook for getting custom data from another module.
- virtual [AAX_Result SetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, const void *i↔ Data)=0
An optional interface hook for setting custom data for use by another module.

MIDI methods

- virtual [AAX_Result DoMIDITransfers](#) ()=0
MIDI update callback.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.63.2 Member Function Documentation

14.63.2.1 Initialize()

```
virtual AAX\_Result AAX_IACFEffParameters::Initialize (
    IACFUnknown * iController ) [pure virtual]
```

Main data model initialization. Called when plug-in instance is first instantiated.

Note

Most plug-ins should override [AAX_CEffectParameters::EffectInit\(\)](#) rather than directly overriding this method

Parameters

in	<i>iController</i>	A versioned reference that resolves to an AAX_IController interface
----	--------------------	---

Implemented in [AAX_CEffectParameters](#).

14.63.2.2 Uninitialize()

```
virtual AAX\_Result AAX_IACFEffParameters::Uninitialize ( ) [pure virtual]
```

Main data model uninitialization.

Todo Docs: When exactly is [AAX_IACFEffParameters::Uninitialize\(\)](#) called, and under what conditions?

Implemented in [AAX_CEffectParameters](#).

14.63.2.3 NotificationReceived()

```
virtual AAX_Result AAX_IACEffectParameters::NotificationReceived (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [pure virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- some notifications are sent only to the plug-in GUI while other notifications are sent only to the plug-in data model. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

A plug-in may also dispatch custom notifications using [AAX_IController::SendNotification\(\)](#). Custom notifications will be posted back to the plug-in's other objects which support a [NotificationReceived\(\)](#) method (e.g. the GUI).

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implemented in [AAX_CEffectParameters](#).

14.63.2.4 GetNumberOfParameters()

```
virtual AAX_Result AAX_IACEffectParameters::GetNumberOfParameters (
    int32_t * oNumControls ) const [pure virtual]
```

CALL: Retrieves the total number of plug-in parameters.

Parameters

out	<i>oNumControls</i>	The number of parameters in the plug-in's Parameter Manager
-----	---------------------	---

Implemented in [AAX_CEffectParameters](#).

14.63.2.5 GetMasterBypassParameter()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetMasterBypassParameter (
    AAX_IString * oIDString ) const [pure virtual]
```

CALL: Retrieves the ID of the plug-in's Master Bypass parameter.

This is required if you want our master bypass functionality in the host to hook up to your bypass parameters.

Parameters

out	<i>oIDString</i>	The ID of the plug-in's Master Bypass control
-----	------------------	---

Implemented in [AAX_CEffectParameters](#).

14.63.2.6 GetParameterIsAutomatable()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetParameterIsAutomatable (
    AAX_CParamID iParameterID,
    AAX_CBoolean * oAutomatable ) const [pure virtual]
```

CALL: Retrieves information about a parameter's automatable status.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oAutomatable</i>	True if the queried parameter is automatable, false if it is not

Implemented in [AAX_CEffectParameters](#).

14.63.2.7 GetParameterNumberOfSteps()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetParameterNumberOfSteps (
    AAX_CParamID iParameterID,
    int32_t * oNumSteps ) const [pure virtual]
```

CALL: Retrieves the number of discrete steps for a parameter.

Note

The value returned for *oNumSteps* MUST be greater than zero. All other values will be considered an error by the host.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oNumSteps</i>	The number of steps for this parameter

Implemented in [AAX_CEffectParameters](#).

14.63.2.8 GetParameterName()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameterName (
    AAX_CParamID iParameterID,
    AAX_IString * oName ) const [pure virtual]
```

CALL: Retrieves the full name for a parameter.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oName</i>	Reference to an AAX_IString owned by the host. The plug-in must set this string equal to the parameter's full name.

Implemented in [AAX_CEffectParameters](#).

14.63.2.9 GetParameterNameOfLength()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameterNameOfLength (
    AAX_CParamID iParameterID,
    AAX_IString * oName,
    int32_t iNameLength ) const [pure virtual]
```

CALL: Retrieves an abbreviated name for a parameter.

In general, lengths of 3 through 8 and 31 should be specifically addressed.

Host Compatibility Notes In most cases, the AAX host will call [GetParameterName\(\)](#) or [GetParameterNameOfLength\(\)](#) to retrieve parameter names for display. However, when Pro Tools is retrieving a plug-in name for display on a control surface the XML data stored in the plug-in's page tables will be used in preference to values retrieved from these methods.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oName</i>	Reference to an AAX_IString owned by the host. The plug-in must set this string equal to an abbreviated name for the parameter, using <i>iNameLength</i> characters or fewer.
in	<i>iNameLength</i>	The maximum number of characters in <i>oName</i>

Implemented in [AAX_CEffectParameters](#).

14.63.2.10 GetParameterDefaultNormalizedValue()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetParameterDefaultNormalizedValue (
    AAX_CParamID iParameterID,
    double * oValue ) const [pure virtual]
```

CALL: Retrieves default value of a parameter.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValue</i>	The parameter's default value

Implemented in [AAX_CEffectParameters](#).

14.63.2.11 SetParameterDefaultNormalizedValue()

```
virtual AAX_Result AAX_IACFEEffectParameters::SetParameterDefaultNormalizedValue (
    AAX_CParamID iParameterID,
    double iValue ) [pure virtual]
```

CALL: Sets the default value of a parameter.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being updated
out	<i>iValue</i>	The parameter's new default value

Todo THIS IS NOT CALLED FROM HOST. USEFUL FOR INTERNAL USE ONLY?

Implemented in [AAX_CEffectParameters](#).

14.63.2.12 GetParameterType()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetParameterType (
    AAX_CParamID iParameterID,
    AAX_EParameterType * oParameterType ) const [pure virtual]
```

CALL: Retrieves the type of a parameter.

Todo The concept of parameter type needs more documentation

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oParameterType</i>	The parameter's type

Implemented in [AAX_CEffectParameters](#).

14.63.2.13 GetParameterOrientation()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameterOrientation (
    AAX_CParamID iParameterID,
    AAX_EParameterOrientation * oParameterOrientation ) const [pure virtual]
```

CALL: Retrieves the orientation that should be applied to a parameter's controls.

Todo update this documentation

This method allows you to specify the orientation of knob controls that are managed by the host (e.g. knobs on an attached control surface.)

Here is an example override of this method that reverses the orientation of a control for a parameter:

```
// AAX_IParameter* myBackwardsParameter
if (iParameterID == myBackwardsParameter->Identifier())
{
    *oParameterType =
        AAX_eParameterOrientation_BottomMinTopMax |
        AAX_eParameterOrientation_LeftMinRightMax |
        AAX_eParameterOrientation_RotaryWrapMode |
        AAX_eParameterOrientation_RotaryLeftMinRightMax;
}
```

The orientation options are set according to [AAX_EParameterOrientationBits](#)

Legacy Porting Notes [AAX_IEffectParameters::GetParameterOrientation\(\)](#) corresponds to the `GetControlOrientation()` method in the legacy RTAS/TDM SDK.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oParameterOrientation</i>	The orientation of the parameter

Implemented in [AAX_CEffectParameters](#).

14.63.2.14 GetParameter()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameter (
    AAX_CParamID iParameterID,
    AAX_IParameter ** oParameter ) [pure virtual]
```

CALL: Retrieves an arbitrary setting within a parameter.

This is a convenience function for accessing the richer parameter interface from the plug-in's other modules.

Note

This function must not be called by the host; [AAX_IParameter](#) is not safe for passing across the binary boundary with the host!

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oParameter</i>	A pointer to the returned parameter

Implemented in [AAX_CEffectParameters](#).

14.63.2.15 GetParameterIndex()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameterIndex (
    AAX_CParamID iParameterID,
    int32_t * oControlIndex ) const [pure virtual]
```

CALL: Retrieves the index of a parameter.

Although parameters are normally referenced by their AAX_CParamID, each parameter is also associated with a unique numeric index.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oControlIndex</i>	The parameter's numeric index

Implemented in [AAX_CEffectParameters](#).

14.63.2.16 GetParameterIDFromIndex()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameterIDFromIndex (
    int32_t iControlIndex,
    AAX_IString * oParameterIDString ) const [pure virtual]
```

CALL: Retrieves the ID of a parameter.

This method can be used to convert a parameter's unique numeric index to its AAX_CParamID

Parameters

in	<i>iControllIndex</i>	The numeric index of the parameter that is being queried
out	<i>oParameterIDString</i>	Reference to an AAX_IString owned by the host. The plug-in must set this string equal to the parameter's ID.

Implemented in [AAX_CEffectParameters](#).

14.63.2.17 GetParameterValueInfo()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameterValueInfo (
    AAX_CParamID iParameterID,
    int32_t iSelector,
    int32_t * oValue ) const [pure virtual]
```

CALL: Retrieves a property of a parameter.

This is a general purpose query that is specialized based on the value of *iSelector*. The currently supported selector values are described by [AAX_EParameterValueInfoSelector](#). The meaning of *oValue* is dependent upon *iSelector*.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
in	<i>iSelector</i>	The selector of the parameter value to retrieve. See AAX_EParameterValueInfoSelector
out	<i>oValue</i>	The value of the specified parameter

Implemented in [AAX_CEffectParameters](#).

14.63.2.18 GetParameterValueFromString()

```
virtual AAX_Result AAX_IACFEffEffectParameters::GetParameterValueFromString (
    AAX_CParamID iParameterID,
    double * oValue,
    const AAX_IString & iValueString ) const [pure virtual]
```

CALL: Converts a value string to a value.

This method uses the queried parameter's display delegate and taper to convert a `char*` string into its corresponding value. The formatting of *valueString* must be supported by the parameter's display delegate in order for this call to succeed.

Legacy Porting Notes This method corresponds to `CProcess::MapControlStringToVal()` in the RTAS/TDM SDK

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValue</i>	The value associated with valueString
in	<i>iValueString</i>	The formatted value string that will be converted into a value

Implemented in [AAX_CEffectParameters](#).

14.63.2.19 GetParameterStringFromValue()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetParameterStringFromValue (
    AAX_CParamID iParameterID,
    double iValue,
    AAX_IString * oValueString,
    int32_t iMaxLength ) const [pure virtual]
```

CALL: Converts a normalized parameter value into a string representing its corresponding real value.

This method uses the queried parameter's display delegate and taper to convert a normalized value into the corresponding `char*` value string for its real value.

Legacy Porting Notes This method corresponds to `CProcess::MapControlValToString()` in the RTAS/TDM SDK

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
in	<i>iValue</i>	The normalized value that will be converted to a formatted valueString
out	<i>oValueString</i>	The formatted value string associated with value
in	<i>iMaxLength</i>	The maximum length of valueString

Implemented in [AAX_CEffectParameters](#).

14.63.2.20 GetParameterValueString()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetParameterValueString (
    AAX_CParamID iParameterID,
    AAX_IString * oValueString,
    int32_t iMaxLength ) const [pure virtual]
```

CALL: Retrieves the value string associated with a parameter's current value.

This method uses the queried parameter's display delegate and taper to convert its current value into a corresponding `char*` value string.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValueString</i>	The formatted value string associated with the parameter's current value
in	<i>iMaxLength</i>	The maximum length of valueString

Implemented in [AAX_CEffectParameters](#).

14.63.2.21 GetParameterNormalizedValue()

```
virtual AAX_Result AAX_IACEffectParameters::GetParameterNormalizedValue (
    AAX_CParamID iParameterID,
    double * oValuePtr ) const [pure virtual]
```

CALL: Retrieves a parameter's current value.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
out	<i>oValuePtr</i>	The parameter's current value

Implemented in [AAX_CEffectParameters](#).

14.63.2.22 SetParameterNormalizedValue()

```
virtual AAX_Result AAX_IACEffectParameters::SetParameterNormalizedValue (
    AAX_CParamID iParameterID,
    double iValue ) [pure virtual]
```

CALL: Sets the specified parameter to a new value.

[SetParameterNormalizedValue\(\)](#) is responsible for initiating any process that is required in order to update all of the parameter's controls (e.g. in the plug-in's GUI, on control surfaces, in automation lanes, etc.) In most cases, the parameter manager will handle this initiation step.

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being set
in	<i>iValue</i>	The value to which the parameter should be set

Implemented in [AAX_CEffectParameters](#).

14.63.2.23 SetParameterNormalizedRelative()

```
virtual AAX_Result AAX_IACFEEffectParameters::SetParameterNormalizedRelative (
    AAX_CParamID iParameterID,
    double iValue ) [pure virtual]
```

CALL: Sets the specified parameter to a new value relative to its current value.

This method is used in cases when a relative control value is more convenient, for example when updating a GUI control using a mouse wheel or the arrow keys. Note that the host may apply the parameter's step size prior to calling [SetParameterNormalizedRelative\(\)](#) in order to determine the correct value for aValue.

[SetParameterNormalizedRelative\(\)](#) can be used to incorporate "wrapping" behavior in a parameter's controls, if desired. If this behavior is not desired, then this method must properly account for overflow of the parameter's normalized value.

[SetParameterNormalizedRelative\(\)](#) is responsible for initiating any process that is required in order to update all of the parameter's controls (e.g. in the plug-in's GUI, on control surfaces, in automation lanes, etc.) In most cases, the parameter manager will handle this initiation step.

See also [UpdateParameterNormalizedRelative\(\)](#).

Todo REMOVE THIS METHOD (?)

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being queried
in	<i>iValue</i>	The change in value that should be applied to the parameter

Todo NOT CURRENTLY CALLED FROM THE HOST. USEFUL FOR INTERNAL USE ONLY?

Implemented in [AAX_CEffectParameters](#).

14.63.2.24 TouchParameter()

```
virtual AAX_Result AAX_IACFEEffectParameters::TouchParameter (
    AAX_CParamID iParameterID ) [pure virtual]
```

"Touches" (locks) a parameter in the automation system to a particular control in preparation for updates

This method is called by the Parameter Manager to prime a parameter for receiving new automation data. When an automatable parameter is touched by a control, it will reject input from other controls until it is released.

Note

You should never need to override this method when using [AAX_CEffectParameters](#).

Parameters

in	<i>iParameterID</i>	The parameter that is being touched
----	---------------------	-------------------------------------

Implemented in [AAX_CEffectParameters](#).

14.63.2.25 ReleaseParameter()

```
virtual AAX_Result AAX_IACEffectParameters::ReleaseParameter (
    AAX_CParamID iParameterID ) [pure virtual]
```

Releases a parameter from a "touched" state.

This method is called by the Parameter Manager to release a parameter so that any control may send updates to the parameter.

Note

You should never need to override this method when using [AAX_CEffectParameters](#).

Parameters

in	<i>iParameterID</i>	The parameter that is being released
----	---------------------	--------------------------------------

Implemented in [AAX_CEffectParameters](#).

14.63.2.26 UpdateParameterTouch()

```
virtual AAX_Result AAX_IACEffectParameters::UpdateParameterTouch (
    AAX_CParamID iParameterID,
    AAX_CBoolean iTouchState ) [pure virtual]
```

Sets a "touched" state on a parameter.

Note

This method should be overridden when dealing with linked parameters. Do NOT use this method to keep track of touch states. Use the [automation delegate](#) for that.

Parameters

in	<i>iParameterID</i>	The parameter that is changing touch states.
in	<i>iTouchState</i>	The touch state of the parameter.

Implemented in [AAX_CEffectParameters](#).

14.63.2.27 UpdateParameterNormalizedValue()

```
virtual AAX_Result AAX_IACFEffEffectParameters::UpdateParameterNormalizedValue (
    AAX_CParamID iParameterID,
    double iValue,
    AAX_EUpdateSource iSource ) [pure virtual]
```

Updates a single parameter's state to its current value.

Note

Do *not* call this method from the plug-in. This method should be called by the host only. To set parameter values from within the plug-in, use the [AAX_IParameter](#) interface.

Todo FLAGGED FOR CONSIDERATION OF REVISION

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being updated
in	<i>iValue</i>	The parameter's current value, to which its internal state must be updated
in	<i>iSource</i>	The source of the update

Implemented in [AAX_CEffectParameters](#), and [AAX_CMonolithicParameters](#).

14.63.2.28 UpdateParameterNormalizedRelative()

```
virtual AAX_Result AAX_IACFEffEffectParameters::UpdateParameterNormalizedRelative (
    AAX_CParamID iParameterID,
    double iValue ) [pure virtual]
```

Updates a single parameter's state to its current value, as a difference with the parameter's previous value.

Deprecated This is not called from the host. It *may* still be useful for internal calls within the plug-in, though it should only ever be used to update non-automatable parameters. Automatable parameters should always be updated through the [AAX_IParameter](#) interface, which will ensure proper coordination with other automation clients.

[UpdateParameterNormalizedRelative\(\)](#) can be used to incorporate "wraparound" behavior in a parameter's controls, if desired. If this behavior is not desired, then this method must properly account for overflow of the parameter's normalized value.

See also

[SetParameterNormalizedRelative\(\)](#)

Parameters

in	<i>iParameterID</i>	The ID of the parameter that is being updated
in	<i>iValue</i>	The difference between the parameter's current value and its previous value (normalized). The parameter's state must be updated to reflect this difference.

Implemented in [AAX_CEffectParameters](#).

14.63.2.29 GenerateCoefficients()

```
virtual AAX_Result AAX_IACEffectParameters::GenerateCoefficients ( ) [pure virtual]
```

Generates and dispatches new coefficient packets.

This method is responsible for updating the coefficient packets associated with all parameters whose states have changed since the last call to [GenerateCoefficients\(\)](#). The host may call this method once for every parameter update, or it may "batch" parameter updates such that changes for several parameters are all handled by a single call to [GenerateCoefficients\(\)](#).

For more information on tracking parameters' statuses using the [AAX_CPacketDispatcher](#), helper class, see [AAX_CPacketDispatcher::SetDirty\(\)](#).

Note

Do *not* call this method from the plug-in. This method should be called by the host only. To set parameter values from within the plug-in, use the [AAX_IParameter](#) interface.

Implemented in [AAX_CMonolithicParameters](#), and [AAX_CEffectParameters](#).

14.63.2.30 ResetFieldData()

```
virtual AAX_Result AAX_IACEffectParameters::ResetFieldData (
    AAX_CFieldIndex inFieldIndex,
    void * oData,
    uint32_t inDataSize ) const [pure virtual]
```

Called by the host to reset a private data field in the plug-in's algorithm.

This method is called sequentially for all private data fields on Effect initialization and during any "reset" event, such as priming for a non-real-time render. This method is called before the algorithm's optional initialization callback, and the initialized private data will be available within that callback via its context block.

See also

[Algorithm initialization](#).

Warning

Any data structures that will be passed between platforms (for example, sent to a TI DSP in an AAX DSP plug-in) must be properly data-aligned for compatibility across both platforms. See [AAX_ALIGN_FILE_ALG](#) for more information about guaranteeing cross-platform compatibility of data structures used for algorithm processing.

Parameters

in	<i>inFieldIndex</i>	The index of the field that is being initialized
out	<i>oData</i>	The pre-allocated block of data that should be initialized
in	<i>inDataSize</i>	The size of the data block, in bytes

Implemented in [AAX_CMonolithicParameters](#), and [AAX_CEffectParameters](#).

14.63.2.31 GetNumberOfChunks()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetNumberOfChunks (
    int32_t * oNumChunks ) const [pure virtual]
```

Retrieves the number of chunks used by this plug-in.

Parameters

out	<i>oNumChunks</i>	The number of distinct chunks used by this plug-in
-----	-------------------	--

Implemented in [AAX_CEffectParameters](#).

14.63.2.32 GetChunkIDFromIndex()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetChunkIDFromIndex (
    int32_t iIndex,
    AAX_CTypeID * oChunkID ) const [pure virtual]
```

Retrieves the ID associated with a chunk index.

Parameters

in	<i>iIndex</i>	Index of the queried chunk
out	<i>oChunkID</i>	ID of the queried chunk

Implemented in [AAX_CEffectParameters](#).

14.63.2.33 GetChunkSize()

```
virtual AAX_Result AAX_IACFEEffectParameters::GetChunkSize (
    AAX_CTypeID iChunkID,
    uint32_t * oSize ) const [pure virtual]
```

Get the size of the data structure that can hold all of a chunk's information.

If *chunkID* is one of the plug-in's custom chunks, initialize **size* to the size of the chunk's data in bytes.

This method is invoked every time a chunk is saved, therefore it is possible to have dynamically sized chunks. However, note that each call to [GetChunkSize\(\)](#) will correspond to a following call to [GetChunk\(\)](#). The chunk provided in [GetChunk\(\)](#) *must* have the same size as the *size* provided by [GetChunkSize\(\)](#).

Legacy Porting Notes In *AAX*, the value provided by [GetChunkSize\(\)](#) should *NOT* include the size of the chunk header. The value should *ONLY* reflect the size of the chunk's data.

Parameters

in	<i>iChunkID</i>	ID of the queried chunk
out	<i>oSize</i>	The chunk's size in bytes

Implemented in [AAX_CEffectParameters](#).

14.63.2.34 GetChunk()

```
virtual AAX_Result AAX_IACEffectParameters::GetChunk (
    AAX_CTypeID iChunkID,
    AAX_SPlugInChunk * oChunk ) const [pure virtual]
```

Fills a block of data with chunk information representing the plug-in's current state.

By calling this method, the host is requesting information about the current state of the plug-in. The following chunk fields should be explicitly populated in this method. Other fields will be populated by the host.

- [AAX_SPlugInChunk::fData](#)
- [AAX_SPlugInChunk::fVersion](#)
- [AAX_SPlugInChunk::fName](#) (Optional)
- [AAX_SPlugInChunk::fSize](#) (Data size only)

Warning

Remember that this chunk data may be loaded on a different platform from the one where it is saved. All data structures in the chunk must be properly data-aligned for compatibility across all platforms that the plug-in supports. See [AAX_ALIGN_FILE_ALG](#) for notes about common cross-platform pitfalls for data structure alignment.

Parameters

in	<i>iChunkID</i>	ID of the chunk that should be provided
out	<i>oChunk</i>	A preallocated block of memory that should be populated with the chunk's data.

Implemented in [AAX_CEffectParameters](#).

14.63.2.35 SetChunk()

```
virtual AAX_Result AAX_IACFEffectParameters::SetChunk (
    AAX_CTypeID iChunkID,
    const AAX_SPlugInChunk * iChunk ) [pure virtual]
```

Restores a set of plug-in parameters based on chunk information.

By calling this method, the host is attempting to update the plug-in's current state to match the data stored in a chunk. The plug-in should initialize itself to this new state by calling [SetParameterNormalizedValue\(\)](#) for each of the relevant parameters.

Parameters

in	<i>iChunkID</i>	ID of the chunk that is being set
in	<i>iChunk</i>	The chunk

Implemented in [AAX_CEffectParameters](#).

14.63.2.36 CompareActiveChunk()

```
virtual AAX_Result AAX_IACFEffectParameters::CompareActiveChunk (
    const AAX_SPlugInChunk * iChunkP,
    AAX_CBoolean * oIsEqual ) const [pure virtual]
```

Determine if a chunk represents settings that are equivalent to the plug-in's current state.

Host Compatibility Notes In Pro Tools, this method will only be called if a prior call to [GetNumberOfChanges\(\)](#) has indicated that the plug-in's state has changed. If the plug-in's current settings are different from the settings in *aChunkP* then the plug-in's Compare Light will be illuminated in the plug-in header, allowing users to toggle between the plug-in's custom state and its saved state.

Parameters

in	<i>iChunkP</i>	The chunk that is to be tested
out	<i>oIsEqual</i>	True if the chunk represents equivalent settings when compared with the plug-in's current state. False if the chunk represents non-equivalent settings

Implemented in [AAX_CEffectParameters](#).

14.63.2.37 GetNumberOfChanges()

```
virtual AAX_Result AAX_IACEffectParameters::GetNumberOfChanges (
    int32_t * oNumChanges ) const [pure virtual]
```

Retrieves the number of parameter changes made since the plug-in's creation.

This method is polled regularly by the host, and can additionally be triggered by some events such as mouse clicks. When the number provided by this method changes, the host subsequently calls [CompareActiveChunk\(\)](#) to determine if the plug-in's Compare light should be activated.

The value provided by this method should increment with each call to [UpdateParameterNormalizedValue\(\)](#)

Parameters

out	<i>oNumChanges</i>	Must be set to indicate the number of parameter changes that have occurred since plug-in initialization.
-----	--------------------	--

Implemented in [AAX_CEffectParameters](#).

14.63.2.38 TimerWakeup()

```
virtual AAX_Result AAX_IACEffectParameters::TimerWakeup ( ) [pure virtual]
```

Periodic wakeup callback for idle-time operations.

This method is called from the host using a non-main thread. In general, it should be driven at approximately one call per 30 ms. However, the wakeup is not guaranteed to be called at any regular interval - for example, it could be held off by a high real-time processing load - and there is no host contract regarding maximum latency between wakeup calls.

This wakeup thread runs continuously and cannot be armed/disarmed or by the plug-in.

Implemented in [AAX_CMonolithicParameters](#), and [AAX_CEffectParameters](#).

14.63.2.39 GetCustomData()

```
virtual AAX_Result AAX_IACEffectParameters::GetCustomData (
    AAX_CTypeID iDataBlockID,
    uint32_t inDataSize,
    void * oData,
    uint32_t * oDataWritten ) const [pure virtual]
```

An optional interface hook for getting custom data from another module.

Parameters

in	<i>iDataBlockID</i>	Identifier for the requested block of custom data
in	<i>inDataSize</i>	Size of provided buffer, in bytes
Generated by Doxygen	<i>oData</i>	Pointer to an allocated buffer. Data will be written here.
out	<i>oDataWritten</i>	The number of bytes actually written

Implemented in [AAX_CEffectParameters](#).

14.63.2.40 SetCustomData()

```
virtual AAX_Result AAX_IACFEffEffectParameters::SetCustomData (
    AAX_CTypeID iDataBlockID,
    uint32_t inDataSize,
    const void * iData ) [pure virtual]
```

An optional interface hook for setting custom data for use by another module.

Parameters

in	<i>iDataBlockID</i>	Identifier for the provided block of custom data
in	<i>inDataSize</i>	Size of provided buffer, in bytes
in	<i>iData</i>	Pointer to the data buffer

Implemented in [AAX_CEffectParameters](#).

14.63.2.41 DoMIDITransfers()

```
virtual AAX_Result AAX_IACFEffEffectParameters::DoMIDITransfers ( ) [pure virtual]
```

MIDI update callback.

Call [AAX_IController::GetNextMIDIPacket\(\)](#) from within this method to retrieve and process MIDI packets directly within the Effect's data model. MIDI data will also be delivered to the Effect algorithm.

This method is called regularly by the host, similarly to [AAX_IEffectParameters::TimerWakeup\(\)](#)

Implemented in [AAX_CEffectParameters](#).

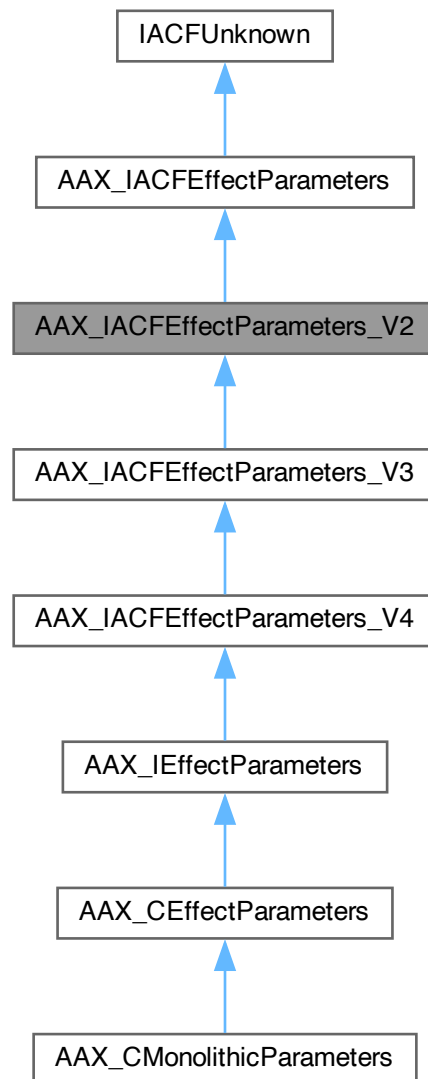
The documentation for this class was generated from the following file:

- [AAX_IACFEffEffectParameters.h](#)

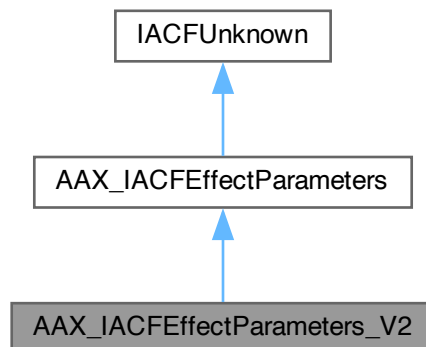
14.64 AAX_IACFEffEffectParameters_V2 Class Reference

```
#include <AAX_IACFEffEffectParameters.h>
```


Inheritance diagram for AAX_IACFEffParameters_V2:



Collaboration diagram for AAX_IACFEffEffectParameters_V2:



14.64.1 Description

Supplemental interface for an AAX Plug-in's data model.

This is a supplemental interface for an instance of a plug-in's data model. This interface gets exposed to the host application. Host applications that support AAX versioned features may call into these methods. See [Data model interface](#).

Note

Your implementation of this interface must inherit from [AAX_IEffectParameters](#).

Todo Add documentation for expected error state return values

Public Member Functions

Hybrid audio methods

- virtual [AAX_Result RenderAudio_Hybrid](#) ([AAX_SHybridRenderInfo](#) *ioRenderInfo)=0
Hybrid audio render function.

MIDI methods

- virtual [AAX_Result UpdateMIDINodes](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_CMidiPacket](#) &iPacket)=0
MIDI update callback.
- virtual [AAX_Result UpdateControlMIDINodes](#) ([AAX_CTypeID](#) nodeID, [AAX_CMidiPacket](#) &iPacket)=0
MIDI update callback for control MIDI nodes.

Public Member Functions inherited from AAX_IACFEffEffectParameters

- virtual [AAX_Result Initialize](#) (IACFUnknown *iController)=0
Main data model initialization. Called when plug-in instance is first instantiated.
- virtual [AAX_Result Uninitialize](#) ()=0
Main data model uninitialization.
- virtual [AAX_Result NotificationReceived](#) (AAX_CTypeID inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.
- virtual [AAX_Result GetNumberOfParameters](#) (int32_t *oNumControls) const =0
CALL: Retrieves the total number of plug-in parameters.
- virtual [AAX_Result GetMasterBypassParameter](#) (AAX_IString *oIDString) const =0
CALL: Retrieves the ID of the plug-in's Master Bypass parameter.
- virtual [AAX_Result GetParameterIsAutomatable](#) (AAX_CParamID iParameterID, AAX_CBoolean *oAutomatable) const =0
CALL: Retrieves information about a parameter's automatable status.
- virtual [AAX_Result GetParameterNumberOfSteps](#) (AAX_CParamID iParameterID, int32_t *oNumSteps) const =0
CALL: Retrieves the number of discrete steps for a parameter.
- virtual [AAX_Result GetParameterName](#) (AAX_CParamID iParameterID, AAX_IString *oName) const =0
CALL: Retrieves the full name for a parameter.
- virtual [AAX_Result GetParameterNameOfLength](#) (AAX_CParamID iParameterID, AAX_IString *oName, int32_t iNameLength) const =0
CALL: Retrieves an abbreviated name for a parameter.
- virtual [AAX_Result GetParameterDefaultNormalizedValue](#) (AAX_CParamID iParameterID, double *oValue) const =0
CALL: Retrieves default value of a parameter.
- virtual [AAX_Result SetParameterDefaultNormalizedValue](#) (AAX_CParamID iParameterID, double iValue)=0
CALL: Sets the default value of a parameter.
- virtual [AAX_Result GetParameterType](#) (AAX_CParamID iParameterID, AAX_EParameterType *oParameterType) const =0
CALL: Retrieves the type of a parameter.
- virtual [AAX_Result GetParameterOrientation](#) (AAX_CParamID iParameterID, AAX_EParameterOrientation *oParameterOrientation) const =0
CALL: Retrieves the orientation that should be applied to a parameter's controls.
- virtual [AAX_Result GetParameter](#) (AAX_CParamID iParameterID, AAX_IParameter **oParameter)=0
CALL: Retrieves an arbitrary setting within a parameter.
- virtual [AAX_Result GetParameterIndex](#) (AAX_CParamID iParameterID, int32_t *oControlIndex) const =0
CALL: Retrieves the index of a parameter.
- virtual [AAX_Result GetParameterIDFromIndex](#) (int32_t iControlIndex, AAX_IString *oParameterIDString) const =0
CALL: Retrieves the ID of a parameter.
- virtual [AAX_Result GetParameterValueInfo](#) (AAX_CParamID iParameterID, int32_t iSelector, int32_t *oValue) const =0
CALL: Retrieves a property of a parameter.
- virtual [AAX_Result GetParameterValueFromString](#) (AAX_CParamID iParameterID, double *oValue, const AAX_IString &iValueString) const =0
CALL: Converts a value string to a value.

- virtual [AAX_Result](#) [GetParameterStringFromValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_IString](#) *oValueString, int32_t iMaxLength) const =0
CALL: Converts a normalized parameter value into a string representing its corresponding real value.
- virtual [AAX_Result](#) [GetParameterValueString](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oValueString, int32_t iMaxLength) const =0
CALL: Retrieves the value string associated with a parameter's current value.
- virtual [AAX_Result](#) [GetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double *oValuePtr) const =0
CALL: Retrieves a parameter's current value.
- virtual [AAX_Result](#) [SetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
CALL: Sets the specified parameter to a new value.
- virtual [AAX_Result](#) [SetParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
CALL: Sets the specified parameter to a new value relative to its current value.

- virtual [AAX_Result](#) [TouchParameter](#) ([AAX_CParamID](#) iParameterID)=0
"Touches" (locks) a parameter in the automation system to a particular control in preparation for updates
- virtual [AAX_Result](#) [ReleaseParameter](#) ([AAX_CParamID](#) iParameterID)=0
Releases a parameter from a "touched" state.
- virtual [AAX_Result](#) [UpdateParameterTouch](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iTouchState)=0
Sets a "touched" state on a parameter.

- virtual [AAX_Result](#) [UpdateParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_EUpdateSource](#) iSource)=0
Updates a single parameter's state to its current value.
- virtual [AAX_Result](#) [UpdateParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
Updates a single parameter's state to its current value, as a difference with the parameter's previous value.
- virtual [AAX_Result](#) [GenerateCoefficients](#) ()=0
Generates and dispatches new coefficient packets.

- virtual [AAX_Result](#) [ResetFieldData](#) ([AAX_CFieldIndex](#) inFieldIndex, void *oData, uint32_t inDataSize) const =0
Called by the host to reset a private data field in the plug-in's algorithm.

- virtual [AAX_Result](#) [GetNumberOfChunks](#) (int32_t *oNumChunks) const =0
Retrieves the number of chunks used by this plug-in.
- virtual [AAX_Result](#) [GetChunkIDFromIndex](#) (int32_t iIndex, [AAX_CTypeID](#) *oChunkID) const =0
Retrieves the ID associated with a chunk index.
- virtual [AAX_Result](#) [GetChunkSize](#) ([AAX_CTypeID](#) iChunkID, uint32_t *oSize) const =0
Get the size of the data structure that can hold all of a chunk's information.
- virtual [AAX_Result](#) [GetChunk](#) ([AAX_CTypeID](#) iChunkID, [AAX_SPlugInChunk](#) *oChunk) const =0
Fills a block of data with chunk information representing the plug-in's current state.
- virtual [AAX_Result](#) [SetChunk](#) ([AAX_CTypeID](#) iChunkID, const [AAX_SPlugInChunk](#) *iChunk)=0
Restores a set of plug-in parameters based on chunk information.
- virtual [AAX_Result](#) [CompareActiveChunk](#) (const [AAX_SPlugInChunk](#) *iChunkP, [AAX_CBoolean](#) *oIsEqual) const =0
Determine if a chunk represents settings that are equivalent to the plug-in's current state.
- virtual [AAX_Result](#) [GetNumberOfChanges](#) (int32_t *oNumChanges) const =0
Retrieves the number of parameter changes made since the plug-in's creation.

- virtual [AAX_Result TimerWakeup](#) ()=0
Periodic wakeup callback for idle-time operations.
- virtual [AAX_Result GetCurveData](#) ([AAX_CTypeID](#) iCurveType, const float *iValues, uint32_t iNumValues, float *oValues) const =0
Generate a set of output values based on a set of given input values.
- virtual [AAX_Result GetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, void *oData, uint32_t *oDataWritten) const =0
An optional interface hook for getting custom data from another module.
- virtual [AAX_Result SetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, const void *iData)=0
An optional interface hook for setting custom data for use by another module.
- virtual [AAX_Result DoMIDITransfers](#) ()=0
MIDI update callback.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.64.2 Member Function Documentation

14.64.2.1 UpdateMIDINodes()

```
virtual AAX\_Result AAX_IACFEffEffectParameters_V2::UpdateMIDINodes (
    AAX\_CFieldIndex inFieldIndex,
    AAX\_CMidiPacket & iPacket ) [pure virtual]
```

MIDI update callback.

This method is called by the host for each pending MIDI packet for MIDI nodes in algorithm context structure. Overwrite this method in Plug-In's EffectParameter class if you want to receive MIDI data packets directly in the data model. MIDI data will also be delivered to the Effect algorithm.

The host calls this method in Effects that register one or more MIDI nodes using [AAX_IComponentDescriptor::AddMIDINode\(\)](#). Effects that do not require MIDI data to be sent to the plug-in algorithm should override [UpdateControlMIDINodes\(\)](#).

Parameters

in	<i>inFieldIndex</i>	MIDI node field index in algorithm context structure
in	<i>iPacket</i>	The incoming MIDI packet for the node

Implemented in [AAX_CEffectParameters](#).

14.64.2.2 UpdateControlMIDINodes()

```
virtual AAX_Result AAX_IACFEffEffectParameters_V2::UpdateControlMIDINodes (
    AAX_CTypeID nodeID,
    AAX_CMidiPacket & iPacket ) [pure virtual]
```

MIDI update callback for control MIDI nodes.

This method is called by the host for each pending MIDI packet for Control MIDI nodes. Overwrite this method in Plug-In's EffectParameter class if you want to receive MIDI data packets directly in the data model.

The host calls this method in Effects that register one or more Control MIDI nodes using [AAX_IEffectDescriptor::AddControlMIDINode\(\)](#). Effects with algorithms that use MIDI data nodes should override [UpdateMIDINodes\(\)](#).

Note

This method will not be called if an Effect includes any MIDI nodes in its algorithm context structure.

Parameters

in	<i>nodeID</i>	Identifier for the MIDI node
in	<i>iPacket</i>	The incoming MIDI packet for the node

Implemented in [AAX_CEffectParameters](#).

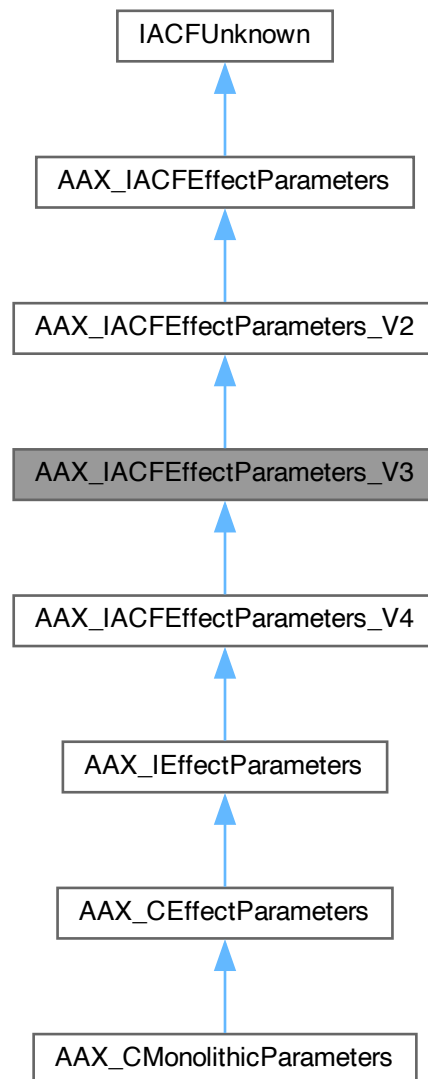
The documentation for this class was generated from the following file:

- [AAX_IACFEffEffectParameters.h](#)

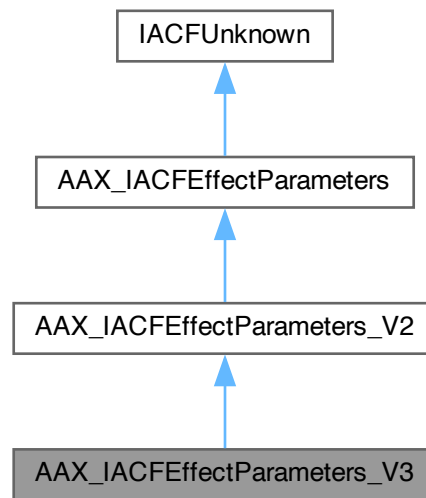
14.65 AAX_IACFEffEffectParameters_V3 Class Reference

```
#include <AAX_IACFEffEffectParameters.h>
```

Inheritance diagram for AAX_IACFEffParameters_V3:



Collaboration diagram for AAX_IACFEffEffectParameters_V3:



14.65.1 Description

Supplemental interface for an AAX Plug-in's data model.

This is a supplemental interface for an instance of a plug-in's data model. This interface gets exposed to the host application. Host applications that support AAX versioned features may call into these methods. See [Data model interface](#).

Note

Your implementation of this interface must inherit from [AAX_IEffectParameters](#).

Todo Add documentation for expected error state return values

Public Member Functions

Auxiliary UI methods

- virtual [AAX_Result](#) [GetCurveDataMeterIds](#) ([AAX_CTypeID](#) iCurveType, uint32_t *oXMeterId, uint32_t *oYMeterId) const =0
Indicates which meters correspond to the X and Y axes of the EQ or Dynamics graph.
- virtual [AAX_Result](#) [GetCurveDataDisplayRange](#) ([AAX_CTypeID](#) iCurveType, float *oXMin, float *oXMax, float *oYMin, float *oYMax) const =0
Determines the range of the graph shown by the plug-in.

Public Member Functions inherited from AAX_IACFEffEffectParameters_V2

- virtual [AAX_Result RenderAudio_Hybrid](#) ([AAX_SHybridRenderInfo](#) *ioRenderInfo)=0
Hybrid audio render function.
- virtual [AAX_Result UpdateMIDINodes](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_CMidiPacket](#) &iPacket)=0
MIDI update callback.
- virtual [AAX_Result UpdateControlMIDINodes](#) ([AAX_CTypeID](#) nodeID, [AAX_CMidiPacket](#) &iPacket)=0
MIDI update callback for control MIDI nodes.

Public Member Functions inherited from AAX_IACFEffEffectParameters

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Main data model initialization. Called when plug-in instance is first instantiated.
- virtual [AAX_Result Uninitialize](#) ()=0
Main data model uninitialization.
- virtual [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, [uint32_t](#) inNotificationDataSize)=0
Notification Hook.
- virtual [AAX_Result GetNumberOfParameters](#) ([int32_t](#) *oNumControls) const =0
CALL: Retrieves the total number of plug-in parameters.
- virtual [AAX_Result GetMasterBypassParameter](#) ([AAX_IString](#) *oIDString) const =0
CALL: Retrieves the ID of the plug-in's Master Bypass parameter.
- virtual [AAX_Result GetParameterIsAutomatable](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *oAutomatable) const =0
CALL: Retrieves information about a parameter's automatable status.
- virtual [AAX_Result GetParameterNumberOfSteps](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) *oNumSteps) const =0
CALL: Retrieves the number of discrete steps for a parameter.
- virtual [AAX_Result GetParameterName](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName) const =0
CALL: Retrieves the full name for a parameter.
- virtual [AAX_Result GetParameterNameOfLength](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName, [int32_t](#) iNameLength) const =0
CALL: Retrieves an abbreviated name for a parameter.
- virtual [AAX_Result GetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double *oValue) const =0
CALL: Retrieves default value of a parameter.
- virtual [AAX_Result SetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
CALL: Sets the default value of a parameter.
- virtual [AAX_Result GetParameterType](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterType](#) *oParameterType) const =0
CALL: Retrieves the type of a parameter.
- virtual [AAX_Result GetParameterOrientation](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterOrientation](#) *oParameterOrientation) const =0
CALL: Retrieves the orientation that should be applied to a parameter's controls.
- virtual [AAX_Result GetParameter](#) ([AAX_CParamID](#) iParameterID, [AAX_IParameter](#) **oParameter)=0
CALL: Retrieves an arbitrary setting within a parameter.
- virtual [AAX_Result GetParameterIndex](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) *oControlIndex) const =0

- CALL: Retrieves the index of a parameter.*
- virtual [AAX_Result GetParameterIDFromIndex](#) (int32_t iControllIndex, [AAX_IString](#) *oParameterIDString) const =0
- CALL: Retrieves the ID of a parameter.*
- virtual [AAX_Result GetParameterValueInfo](#) ([AAX_CParamID](#) iParameterID, int32_t iSelector, int32_t *oValue) const =0
- CALL: Retrieves a property of a parameter.*
-
- virtual [AAX_Result GetParameterValueFromString](#) ([AAX_CParamID](#) iParameterID, double *oValue, const [AAX_IString](#) &iValueString) const =0
- CALL: Converts a value string to a value.*
- virtual [AAX_Result GetParameterStringFromValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_IString](#) *oValueString, int32_t iMaxLength) const =0
- CALL: Converts a normalized parameter value into a string representing its corresponding real value.*
- virtual [AAX_Result GetParameterValueString](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oValueString, int32_t iMaxLength) const =0
- CALL: Retrieves the value string associated with a parameter's current value.*
- virtual [AAX_Result GetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double *oValuePtr) const =0
- CALL: Retrieves a parameter's current value.*
- virtual [AAX_Result SetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
- CALL: Sets the specified parameter to a new value.*
- virtual [AAX_Result SetParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
- CALL: Sets the specified parameter to a new value relative to its current value.*
-
- virtual [AAX_Result TouchParameter](#) ([AAX_CParamID](#) iParameterID)=0
- "Touches" (locks) a parameter in the automation system to a particular control in preparation for updates*
- virtual [AAX_Result ReleaseParameter](#) ([AAX_CParamID](#) iParameterID)=0
- Releases a parameter from a "touched" state.*
- virtual [AAX_Result UpdateParameterTouch](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iTouchState)=0
- Sets a "touched" state on a parameter.*
-
- virtual [AAX_Result UpdateParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_EUpdateSource](#) iSource)=0
- Updates a single parameter's state to its current value.*
- virtual [AAX_Result UpdateParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
- Updates a single parameter's state to its current value, as a difference with the parameter's previous value.*
- virtual [AAX_Result GenerateCoefficients](#) ()=0
- Generates and dispatches new coefficient packets.*
-
- virtual [AAX_Result ResetFieldData](#) ([AAX_CFieldIndex](#) inFieldIndex, void *oData, uint32_t inDataSize) const =0
- Called by the host to reset a private data field in the plug-in's algorithm.*
-
- virtual [AAX_Result GetNumberOfChunks](#) (int32_t *oNumChunks) const =0
- Retrieves the number of chunks used by this plug-in.*
- virtual [AAX_Result GetChunkIDFromIndex](#) (int32_t iIndex, [AAX_CTypeID](#) *oChunkID) const =0
- Retrieves the ID associated with a chunk index.*
- virtual [AAX_Result GetChunkSize](#) ([AAX_CTypeID](#) iChunkID, uint32_t *oSize) const =0
- Get the size of the data structure that can hold all of a chunk's information.*

- virtual [AAX_Result GetChunk](#) ([AAX_CTypeID](#) iChunkID, [AAX_SPlugInChunk](#) *oChunk) const =0
Fills a block of data with chunk information representing the plug-in's current state.
- virtual [AAX_Result SetChunk](#) ([AAX_CTypeID](#) iChunkID, const [AAX_SPlugInChunk](#) *iChunk)=0
Restores a set of plug-in parameters based on chunk information.
- virtual [AAX_Result CompareActiveChunk](#) (const [AAX_SPlugInChunk](#) *iChunkP, [AAX_CBoolean](#) *oIsEqual) const =0
Determine if a chunk represents settings that are equivalent to the plug-in's current state.
- virtual [AAX_Result GetNumberOfChanges](#) (int32_t *oNumChanges) const =0
Retrieves the number of parameter changes made since the plug-in's creation.
- virtual [AAX_Result TimerWakeup](#) ()=0
Periodic wakeup callback for idle-time operations.
- virtual [AAX_Result GetCurveData](#) ([AAX_CTypeID](#) iCurveType, const float *iValues, uint32_t iNumValues, float *oValues) const =0
Generate a set of output values based on a set of given input values.
- virtual [AAX_Result GetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, void *oData, uint32_t *oDataWritten) const =0
An optional interface hook for getting custom data from another module.
- virtual [AAX_Result SetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, const void *iData)=0
An optional interface hook for setting custom data for use by another module.
- virtual [AAX_Result DoMIDITransfers](#) ()=0
MIDI update callback.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

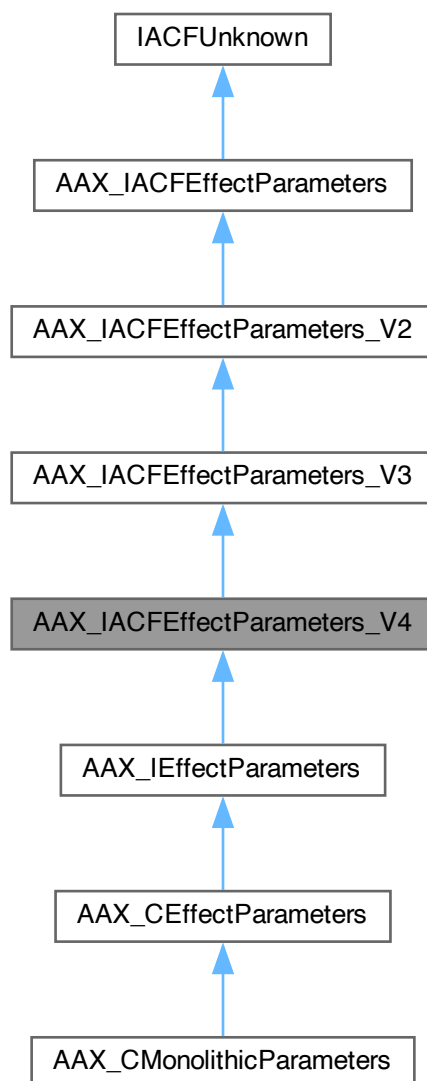
The documentation for this class was generated from the following file:

- [AAX_IACFEffEffectParameters.h](#)

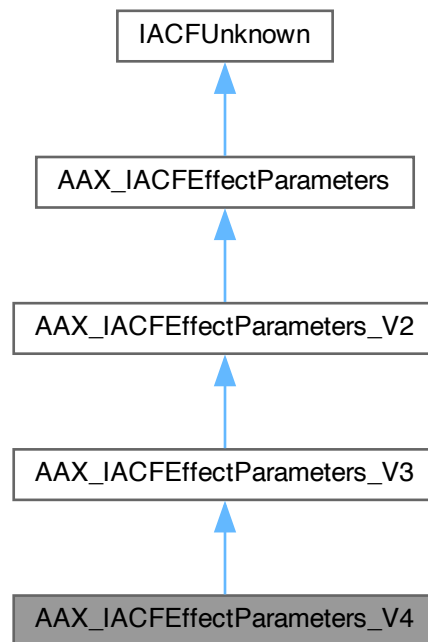
14.66 AAX_IACFEffEffectParameters_V4 Class Reference

```
#include <AAX_IACFEffEffectParameters.h>
```

Inheritance diagram for AAX_IACFEffEffectParameters_V4:



Collaboration diagram for AAX_IACFEffParameters_V4:



14.66.1 Description

Supplemental interface for an AAX Plug-in's data model.

This is a supplemental interface for an instance of a plug-in's data model. This interface gets exposed to the host application. Host applications that support AAX versioned features may call into these methods. See [Data model interface](#).

Note

Your implementation of this interface must inherit from [AAX_IEffectParameters](#).

Todo Add documentation for expected error state return values

Public Member Functions

Auxiliary UI methods

- virtual [AAX_Result UpdatePageTable](#) (uint32_t inTableType, int32_t inTablePageSize, [IACFUnknown](#) *iHostUnknown, [IACFUnknown](#) *ioPageTableUnknown) const =0
Allow the plug-in to update its page tables.

Public Member Functions inherited from [AAX_IACFEffEffectParameters_V3](#)

- virtual [AAX_Result](#) [GetCurveDataMeterIds](#) ([AAX_CTypeID](#) iCurveType, [uint32_t](#) *oXMeterId, [uint32_t](#) *oYMeterId) const =0
Indicates which meters correspond to the X and Y axes of the EQ or Dynamics graph.
- virtual [AAX_Result](#) [GetCurveDataDisplayRange](#) ([AAX_CTypeID](#) iCurveType, [float](#) *oXMin, [float](#) *oXMax, [float](#) *oYMin, [float](#) *oYMax) const =0
Determines the range of the graph shown by the plug-in.

Public Member Functions inherited from [AAX_IACFEffEffectParameters_V2](#)

- virtual [AAX_Result](#) [RenderAudio_Hybrid](#) ([AAX_SHybridRenderInfo](#) *ioRenderInfo)=0
Hybrid audio render function.
- virtual [AAX_Result](#) [UpdateMIDINodes](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_CMidiPacket](#) &iPacket)=0
MIDI update callback.
- virtual [AAX_Result](#) [UpdateControlMIDINodes](#) ([AAX_CTypeID](#) nodeID, [AAX_CMidiPacket](#) &iPacket)=0
MIDI update callback for control MIDI nodes.

Public Member Functions inherited from [AAX_IACFEffEffectParameters](#)

- virtual [AAX_Result](#) [Initialize](#) ([IACFUnknown](#) *iController)=0
Main data model initialization. Called when plug-in instance is first instantiated.
- virtual [AAX_Result](#) [Uninitialize](#) ()=0
Main data model uninitialization.
- virtual [AAX_Result](#) [NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, [const void](#) *inNotificationData, [uint32_t](#) inNotificationDataSize)=0
Notification Hook.
- virtual [AAX_Result](#) [GetNumberOfParameters](#) ([int32_t](#) *oNumControls) const =0
CALL: Retrieves the total number of plug-in parameters.
- virtual [AAX_Result](#) [GetMasterBypassParameter](#) ([AAX_IString](#) *oIDString) const =0
CALL: Retrieves the ID of the plug-in's Master Bypass parameter.
- virtual [AAX_Result](#) [GetParameterIsAutomatable](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *oAutomatable) const =0
CALL: Retrieves information about a parameter's automatable status.
- virtual [AAX_Result](#) [GetParameterNumberOfSteps](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) *oNumSteps) const =0
CALL: Retrieves the number of discrete steps for a parameter.
- virtual [AAX_Result](#) [GetParameterName](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName) const =0
CALL: Retrieves the full name for a parameter.
- virtual [AAX_Result](#) [GetParameterNameOfLength](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName, [int32_t](#) iNameLength) const =0
CALL: Retrieves an abbreviated name for a parameter.
- virtual [AAX_Result](#) [GetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) *oValue) const =0
CALL: Retrieves default value of a parameter.
- virtual [AAX_Result](#) [SetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue)=0

- CALL: Sets the default value of a parameter.*
- virtual [AAX_Result](#) [GetParameterType](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterType](#) *oParameterType) const =0
- CALL: Retrieves the type of a parameter.*
- virtual [AAX_Result](#) [GetParameterOrientation](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterOrientation](#) *oParameterOrientation) const =0
- CALL: Retrieves the orientation that should be applied to a parameter's controls.*
- virtual [AAX_Result](#) [GetParameter](#) ([AAX_CParamID](#) iParameterID, [AAX_IParameter](#) **oParameter)=0
- CALL: Retrieves an arbitrary setting within a parameter.*
- virtual [AAX_Result](#) [GetParameterIndex](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) *oControllIndex) const =0
- CALL: Retrieves the index of a parameter.*
- virtual [AAX_Result](#) [GetParameterIDFromIndex](#) ([int32_t](#) iControllIndex, [AAX_IString](#) *oParameterIDString) const =0
- CALL: Retrieves the ID of a parameter.*
- virtual [AAX_Result](#) [GetParameterValueInfo](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) iSelector, [int32_t](#) *oValue) const =0
- CALL: Retrieves a property of a parameter.*
-
- virtual [AAX_Result](#) [GetParameterValueFromString](#) ([AAX_CParamID](#) iParameterID, [double](#) *oValue, const [AAX_IString](#) &iValueString) const =0
- CALL: Converts a value string to a value.*
- virtual [AAX_Result](#) [GetParameterStringFromValue](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const =0
- CALL: Converts a normalized parameter value into a string representing its corresponding real value.*
- virtual [AAX_Result](#) [GetParameterValueString](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const =0
- CALL: Retrieves the value string associated with a parameter's current value.*
- virtual [AAX_Result](#) [GetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) *oValuePtr) const =0
- CALL: Retrieves a parameter's current value.*
- virtual [AAX_Result](#) [SetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue)=0
- CALL: Sets the specified parameter to a new value.*
- virtual [AAX_Result](#) [SetParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue)=0
- CALL: Sets the specified parameter to a new value relative to its current value.*
-
- virtual [AAX_Result](#) [TouchParameter](#) ([AAX_CParamID](#) iParameterID)=0
- "Touches" (locks) a parameter in the automation system to a particular control in preparation for updates*
- virtual [AAX_Result](#) [ReleaseParameter](#) ([AAX_CParamID](#) iParameterID)=0
- Releases a parameter from a "touched" state.*
- virtual [AAX_Result](#) [UpdateParameterTouch](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iTouchState)=0
- Sets a "touched" state on a parameter.*
-
- virtual [AAX_Result](#) [UpdateParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue, [AAX_EUpdateSource](#) iSource)=0
- Updates a single parameter's state to its current value.*
- virtual [AAX_Result](#) [UpdateParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, [double](#) iValue)=0
- Updates a single parameter's state to its current value, as a difference with the parameter's previous value.*
- virtual [AAX_Result](#) [GenerateCoefficients](#) ()=0
- Generates and dispatches new coefficient packets.*

- virtual [AAX_Result ResetFieldData](#) ([AAX_CFieldIndex](#) inFieldIndex, void *oData, uint32_t inDataSize) const =0
Called by the host to reset a private data field in the plug-in's algorithm.
- virtual [AAX_Result GetNumberOfChunks](#) (int32_t *oNumChunks) const =0
Retrieves the number of chunks used by this plug-in.
- virtual [AAX_Result GetChunkIDFromIndex](#) (int32_t iIndex, [AAX_CTypeID](#) *oChunkID) const =0
Retrieves the ID associated with a chunk index.
- virtual [AAX_Result GetChunkSize](#) ([AAX_CTypeID](#) iChunkID, uint32_t *oSize) const =0
Get the size of the data structure that can hold all of a chunk's information.
- virtual [AAX_Result GetChunk](#) ([AAX_CTypeID](#) iChunkID, [AAX_SPlugInChunk](#) *oChunk) const =0
Fills a block of data with chunk information representing the plug-in's current state.
- virtual [AAX_Result SetChunk](#) ([AAX_CTypeID](#) iChunkID, const [AAX_SPlugInChunk](#) *iChunk)=0
Restores a set of plug-in parameters based on chunk information.
- virtual [AAX_Result CompareActiveChunk](#) (const [AAX_SPlugInChunk](#) *iChunkP, [AAX_CBoolean](#) *oIsEqual) const =0
Determine if a chunk represents settings that are equivalent to the plug-in's current state.
- virtual [AAX_Result GetNumberOfChanges](#) (int32_t *oNumChanges) const =0
Retrieves the number of parameter changes made since the plug-in's creation.
- virtual [AAX_Result TimerWakeup](#) ()=0
Periodic wakeup callback for idle-time operations.
- virtual [AAX_Result GetCurveData](#) ([AAX_CTypeID](#) iCurveType, const float *iValues, uint32_t iNumValues, float *oValues) const =0
Generate a set of output values based on a set of given input values.
- virtual [AAX_Result GetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, void *oData, uint32_t *oDataWritten) const =0
An optional interface hook for getting custom data from another module.
- virtual [AAX_Result SetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, const void *iData)=0
An optional interface hook for setting custom data for use by another module.
- virtual [AAX_Result DoMIDITransfers](#) ()=0
MIDI update callback.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.66.2 Member Function Documentation

14.66.2.1 UpdatePageTable()

```
virtual AAX_Result AAX_IACFEffEffectParameters_V4::UpdatePageTable (
    uint32_t inTableType,
    int32_t inTablePageSize,
    IACFUnknown * iHostUnknown,
    IACFUnknown * ioPageTableUnknown ) const [pure virtual]
```

Allow the plug-in to update its page tables.

Called by the plug-in host, usually in response to a [AAX_eNotificationEvent_ParameterMappingChanged](#) notification sent from the plug-in.

Use this method to change the page table mapping for the plug-in instance or to apply other changes to auxiliary UIs which use the plug-in page tables, such as setting focus to a new page.

See [Page Table Guide](#) for more information about page tables.

Parameters

in	<i>inTableType</i>	Four-char type identifier for the table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the table
in	<i>iHostUnknown</i>	Unknown interface from the host which may support interfaces providing additional features or information. All interfaces queried from this unknown will be valid only within the scope of this UpdatePageTable() execution and will be relevant for only the current plug-in instance.
in, out	<i>ioPageTableUnknown</i>	Unknown interface which supports AAX_IPageTable . This object represents the page table data which is currently stored by the host for this plug-in instance for the given table type and page size. This data and may be edited within the scope of UpdatePageTable() to change the page table mapping for this plug-in instance.

Returns

This method should return [AAX_ERROR_UNIMPLEMENTED](#) if the plug-in does not implement it or when no change is requested by the plug-in. This allows optimizations to be used in the host when no UI update is required following this call.

See also

[AAX_eNotificationEvent_ParameterMappingChanged](#)

Implemented in [AAX_CEffectParameters](#).

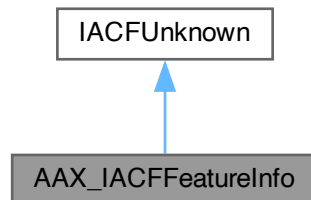
The documentation for this class was generated from the following file:

- [AAX_IACFEffEffectParameters.h](#)

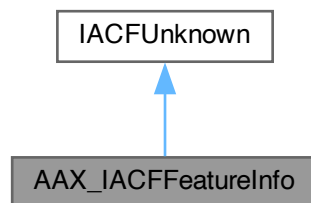
14.67 AAX_IACFFeatureInfo Class Reference

```
#include <AAX_IACFFeatureInfo.h>
```

Inheritance diagram for AAX_IACFFeatureInfo:



Collaboration diagram for AAX_IACFFeatureInfo:



14.67.1 Description

Information about host support for a particular feature

Acquired using [AAX_IACFDescriptionHost::AcquireFeatureProperties\(\)](#)

This interface is shared between multiple features. The specific feature which this object represents is the feature whose ID was used in the call to acquire this interface.

See the feature UID documentation for which properties support additional property map data

IID: [IID_IAAXFeatureInfoV1](#)

Note

Do not [QueryInterface\(\)](#) for [IID_IAAXFeatureInfoV1](#) since this does not indicate which specific feature is being requested. Instead, use [AAX_IDescriptionHost::AcquireFeatureProperties\(\)](#)

Public Member Functions

- virtual [AAX_Result SupportLevel](#) ([AAX_ESupportLevel](#) *oSupportLevel) const =0
- virtual [AAX_Result AcquireProperties](#) ([IACFUnknown](#) **outProperties)=0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.67.2 Member Function Documentation

14.67.2.1 SupportLevel()

```
virtual AAX\_Result AAX_IACFFeatureInfo::SupportLevel (
    AAX\_ESupportLevel * oSupportLevel ) const [pure virtual]
```

Determine the level of support for this feature by the host

Note

The host will not provide an underlying [AAX_IACFFeatureInfo](#) interface for features which it does not recognize at all, resulting in a [AAX_ERROR_NULL_OBJECT](#) error code

See also

[AAX_IFeatureInfo::SupportLevel\(\)](#)

Determine the level of support for this feature by the host

Note

The host will not provide an underlying [AAX_IACFFeatureInfo](#) interface for features which it does not recognize at all, resulting in a [AAX_ERROR_NULL_OBJECT](#) error code

14.67.2.2 AcquireProperties()

```
virtual AAX_Result AAX_IACFFeatureInfo::AcquireProperties (
    IACFUnknown ** outProperties ) [pure virtual]
```

`outProperties` must support [AAX_IACFPropertyMap](#) const methods

See also

[AAX_IFeatureInfo::AcquireProperties\(\)](#)

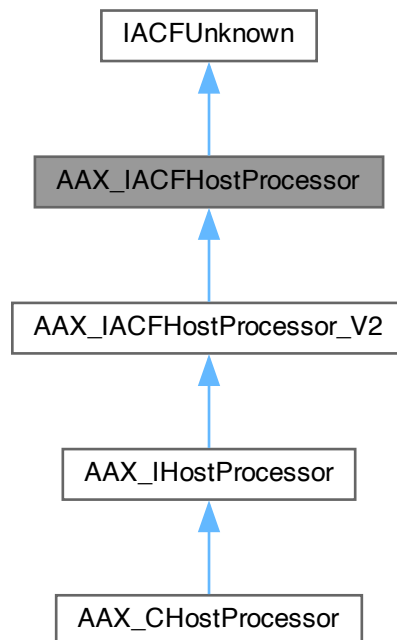
The documentation for this class was generated from the following file:

- [AAX_IACFFeatureInfo.h](#)

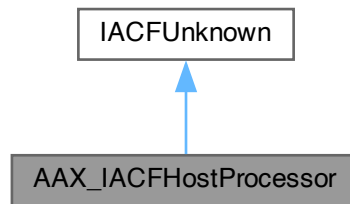
14.68 AAX_IACFHostProcessor Class Reference

```
#include <AAX_IACFHostProcessor.h>
```

Inheritance diagram for `AAX_IACFHostProcessor`:



Collaboration diagram for AAX_IACFHostProcessor:



14.68.1 Description

Versioned interface for an AAX host processing component.

Note

This interface gets exposed to the host application. See [AAX_CHostProcessor](#) for method documentation.

Legacy Porting Notes This interface provides offline processing features analogous to the legacy AudioSuite plug-in architecture

Public Member Functions

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Host Processor initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Host Processor teardown.
- virtual [AAX_Result InitOutputBounds](#) (int64_t iSrcStart, int64_t iSrcEnd, int64_t *oDstStart, int64_t *oDstEnd)=0
Sets the processing region.
- virtual [AAX_Result SetLocation](#) (int64_t iSample)=0
Updates the relative sample location of the current processing frame.
- virtual [AAX_Result RenderAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, float *const iAudioOuts[], int32_t iAudioOutCount, int32_t *ioWindowSize)=0
Perform the signal processing.
- virtual [AAX_Result PreRender](#) (int32_t inAudioInCount, int32_t iAudioOutCount, int32_t iWindowSize)=0
Invoked right before the start of a Preview or Render pass.
- virtual [AAX_Result PostRender](#) ()=0
Invoked at the end of a Render pass.
- virtual [AAX_Result AnalyzeAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int32_t *ioWindowSize)=0
Override this method if the plug-in needs to analyze the audio prior to a Render pass.
- virtual [AAX_Result PreAnalyze](#) (int32_t inAudioInCount, int32_t iWindowSize)=0
Invoked right before the start of an Analysis pass.
- virtual [AAX_Result PostAnalyze](#) ()=0
Invoked at the end of an Analysis pass.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.68.2 Member Function Documentation

14.68.2.1 Initialize()

```
virtual AAX\_Result AAX_IACFHostProcessor::Initialize (
    IACFUnknown * iController ) [pure virtual]
```

Host Processor initialization.

Parameters

in	<i>iController</i>	A versioned reference that can be resolved to both an AAX_IController interface and an AAX_IHostProcessorDelegate
----	--------------------	---

Implemented in [AAX_CHostProcessor](#).

14.68.2.2 Uninitialize()

```
virtual AAX\_Result AAX_IACFHostProcessor::Uninitialize ( ) [pure virtual]
```

Host Processor teardown.

Implemented in [AAX_CHostProcessor](#).

14.68.2.3 InitOutputBounds()

```
virtual AAX\_Result AAX_IACFHostProcessor::InitOutputBounds (
    int64_t iSrcStart,
    int64_t iSrcEnd,
    int64_t * oDstStart,
    int64_t * oDstEnd ) [pure virtual]
```

Sets the processing region.

This method allows offline processing plug-ins to vary the length and/or start/end points of the audio processing region.

This method is called in a few different scenarios:

- Before an analyze, process or preview of data begins.
- At the end of every preview loop.
- After the user makes a new data selection on the timeline.

Plug-ins that inherit from [AAX_CHostProcessor](#) should not override this method. Instead, use the following convenience functions:

- To retrieve the length or boundaries of the processing region, use [GetInputRange\(\)](#), [GetSrcStart\(\)](#), etc.
- To change the boundaries of the processing region before processing begins, use [AAX_CHostProcessor::TranslateOutputBounds\(\)](#).

Note

Currently, a host processor may not randomly access samples outside of the boundary defined by `oDstStart` and `oDstEnd`.

Legacy Porting Notes DAE no longer makes use of the `mStartBound` and `mEndBounds` member variables that existed in the legacy RTAS/TDM SDK. Use `oDstStart` and `oDstEnd` instead (preferably by overriding [TranslateOutputBounds\(\)](#).)

Parameters

in	<i>iSrcStart</i>	The selection start of the user selected region. This will always return 0 for a given selection on the timeline.
in	<i>iSrcEnd</i>	The selection end of the user selected region. This will always return the value of the selection length on the timeline.
in	<i>oDstStart</i>	The starting sample location in the output audio region. By default, this is the same as <code>iSrcStart</code> .
in	<i>oDstEnd</i>	The ending sample location in the output audio region. By default, this is the same as <code>iSrcEnd</code> .

Implemented in [AAX_CHostProcessor](#).

14.68.2.4 SetLocation()

```
virtual AAX\_Result AAX_IACFHostProcessor::SetLocation (
    int64_t iSample ) [pure virtual]
```

Updates the relative sample location of the current processing frame.

This method is called by the host to update the relative sample location of the current processing frame.

Note

Plug-ins should not override this method; instead, use [AAX_CHostProcessor::GetLocation\(\)](#) to retrieve the current relative sample location.

Parameters

in	<i>iSample</i>	The sample location of the first sample in the current processing frame relative to the beginning of the full processing buffer
----	----------------	---

Implemented in [AAX_CHostProcessor](#).

14.68.2.5 RenderAudio()

```
virtual AAX_Result AAX_IACFHostProcessor::RenderAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    float *const iAudioOuts[],
    int32_t iAudioOutCount,
    int32_t * ioWindowSize ) [pure virtual]
```

Perform the signal processing.

This method is called by the host to invoke the plug-in's signal processing.

Legacy Porting Notes This method is a replacement for the AudioSuite `ProcessAudio` method

Parameters

in	<i>inAudioIns</i>	Input audio buffer
in	<i>inAudioInCount</i>	The number if input channels
in	<i>iAudioOuts</i>	The number of output channels
in	<i>iAudioOutCount</i>	A user defined destination end of the ingested audio
in	<i>ioWindowSize</i>	Window buffer length of the received audio

Implemented in [AAX_CHostProcessor](#).

14.68.2.6 PreRender()

```
virtual AAX_Result AAX_IACFHostProcessor::PreRender (
    int32_t inAudioInCount,
    int32_t iAudioOutCount,
    int32_t iWindowSize ) [pure virtual]
```

Invoked right before the start of a Preview or Render pass.

This method is called by the host to allow a plug-in to make any initializations before processing actually begins. Upon a Preview pass, PreRender will also be called at the beginning of every "loop".

See also

[AAX_eProcessingState_StartPass](#), [AAX_eProcessingState_BeginPassGroup](#)

Parameters

in	<i>inAudioInCount</i>	The number if input channels
in	<i>iAudioOutCount</i>	The number of output channels
in	<i>iWindowSize</i>	Window buffer length of the ingested audio

Implemented in [AAX_CHostProcessor](#).

14.68.2.7 PostRender()

```
virtual AAX_Result AAX_IACFHostProcessor::PostRender ( ) [pure virtual]
```

Invoked at the end of a Render pass.

Note

Upon a Preview pass, PostRender will not be called until Preview has stopped.

See also

[AAX_eProcessingState_StopPass](#), [AAX_eProcessingState_EndPassGroup](#)

Implemented in [AAX_CHostProcessor](#).

14.68.2.8 AnalyzeAudio()

```
virtual AAX_Result AAX_IACFHostProcessor::AnalyzeAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    int32_t * ioWindowSize ) [pure virtual]
```

Override this method if the plug-in needs to analyze the audio prior to a Render pass.

Use this after declaring the appropriate properties in Describe. See [AAX_eProperty_RequiresAnalysis](#) and [AAX_eProperty_OptionalAnalysis](#)

To request an analysis pass from within a plug-in, use [AAX_IHostProcessorDelegate::ForceAnalyze\(\)](#)

Legacy Porting Notes Ported from AudioSuite's `AnalyzeAudio(bool isMasterBypassed)` method

Parameters

in	<i>inAudioIns</i>	Input audio buffer
in	<i>inAudioInCount</i>	The number of input channels
in	<i>ioWindowSize</i>	Window buffer length of the ingested audio

Implemented in [AAX_CHostProcessor](#).

14.68.2.9 PreAnalyze()

```
virtual AAX_Result AAX_IACFHostProcessor::PreAnalyze (
    int32_t inAudioInCount,
    int32_t iWindowSize ) [pure virtual]
```

Invoked right before the start of an Analysis pass.

This method is called by the host to allow a plug-in to make any initializations before an Analysis pass actually begins.

See also

[AAX_eProcessingState_StartPass](#), [AAX_eProcessingState_BeginPassGroup](#)

Parameters

in	<i>inAudioInCount</i>	The number if input channels
in	<i>iWindowSize</i>	Window buffer length of the ingested audio

Implemented in [AAX_CHostProcessor](#).

14.68.2.10 PostAnalyze()

```
virtual AAX_Result AAX_IACFHostProcessor::PostAnalyze ( ) [pure virtual]
```

Invoked at the end of an Analysis pass.

Note

In Pro Tools, a long execution time for this method will hold off the main application thread and cause a visible hang. If the plug-in must perform any long running tasks before initiating processing then it is best to perform these tasks in [AAX_IHostProcessor::PreRender\(\)](#)

See also

[AAX_eProcessingState_StopPass](#), [AAX_eProcessingState_EndPassGroup](#)

Implemented in [AAX_CHostProcessor](#).

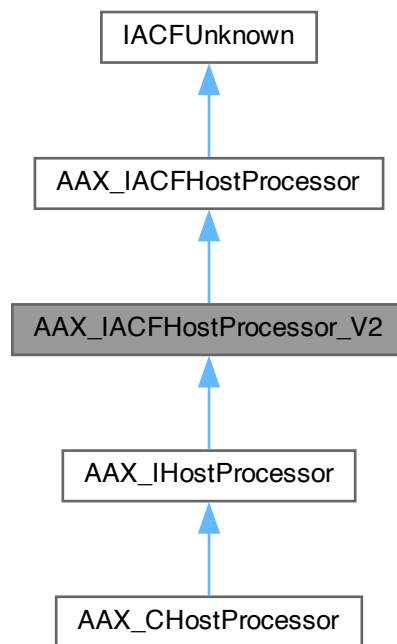
The documentation for this class was generated from the following file:

- [AAX_IACFHostProcessor.h](#)

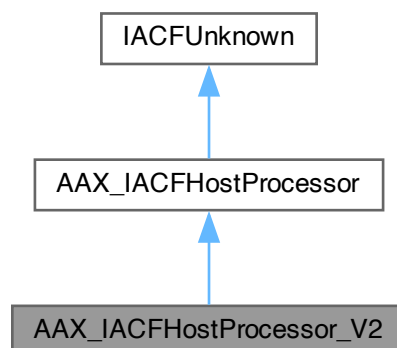
14.69 AAX_IACFHostProcessor_V2 Class Reference

```
#include <AAX_IACFHostProcessor.h>
```

Inheritance diagram for AAX_IACFHostProcessor_V2:



Collaboration diagram for AAX_IACFHostProcessor_V2:



14.69.1 Description

Supplemental interface for an AAX host processing component.

Note

This interface gets exposed to the host application. See [AAX_CHostProcessor](#) for method documentation.

Public Member Functions

- virtual [AAX_Result](#) [GetClipNameSuffix](#) (int32_t inMaxLength, [AAX_IString](#) *outString) const =0
Called by host application to retrieve a custom string to be appended to the clip name.

Public Member Functions inherited from [AAX_IACFHostProcessor](#)

- virtual [AAX_Result](#) [Initialize](#) ([IACFUnknown](#) *iController)=0
Host Processor initialization.
- virtual [AAX_Result](#) [Uninitialize](#) ()=0
Host Processor teardown.
- virtual [AAX_Result](#) [InitOutputBounds](#) (int64_t iSrcStart, int64_t iSrcEnd, int64_t *oDstStart, int64_t *oDstEnd)=0
Sets the processing region.
- virtual [AAX_Result](#) [SetLocation](#) (int64_t iSample)=0
Updates the relative sample location of the current processing frame.
- virtual [AAX_Result](#) [RenderAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, float *const iAudioOuts[], int32_t iAudioOutCount, int32_t *ioWindowSize)=0
Perform the signal processing.
- virtual [AAX_Result](#) [PreRender](#) (int32_t inAudioInCount, int32_t iAudioOutCount, int32_t iWindowSize)=0
Invoked right before the start of a Preview or Render pass.
- virtual [AAX_Result](#) [PostRender](#) ()=0
Invoked at the end of a Render pass.
- virtual [AAX_Result](#) [AnalyzeAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int32_t *ioWindowSize)=0
Override this method if the plug-in needs to analyze the audio prior to a Render pass.
- virtual [AAX_Result](#) [PreAnalyze](#) (int32_t inAudioInCount, int32_t iWindowSize)=0
Invoked right before the start of an Analysis pass.
- virtual [AAX_Result](#) [PostAnalyze](#) ()=0
Invoked at the end of an Analysis pass.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.69.2 Member Function Documentation

14.69.2.1 GetClipNameSuffix()

```
virtual AAX_Result AAX_IACFHostProcessor_V2::GetClipNameSuffix (
    int32_t inMaxLength,
    AAX_IString * outString ) const [pure virtual]
```

Called by host application to retrieve a custom string to be appended to the clip name.

If no string is provided then the host's default will be used.

Parameters

in	<i>inMaxLength</i>	The maximum allowed string length, not including the NULL terminating char
out	<i>outString</i>	Add a value to this string to provide a custom clip suffix

Implemented in [AAX_CHostProcessor](#).

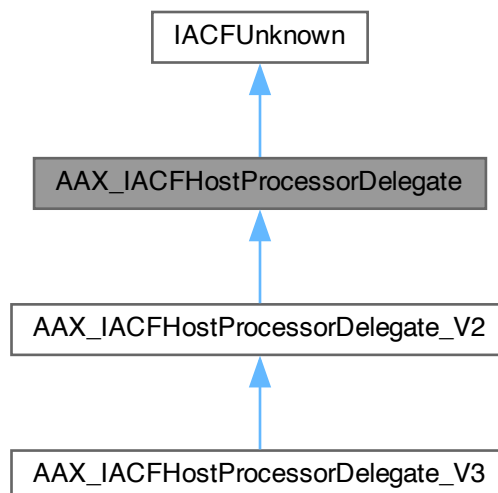
The documentation for this class was generated from the following file:

- [AAX_IACFHostProcessor.h](#)

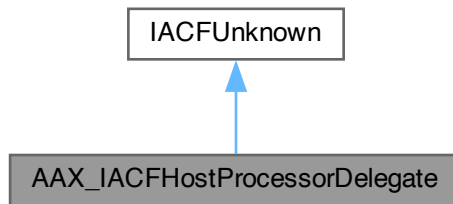
14.70 AAX_IACFHostProcessorDelegate Class Reference

```
#include <AAX_IACFHostProcessorDelegate.h>
```

Inheritance diagram for AAX_IACFHostProcessorDelegate:



Collaboration diagram for AAX_IACFHostProcessorDelegate:



14.70.1 Description

Versioned interface for host methods specific to offline processing.

Public Member Functions

- virtual [AAX_Result](#) [GetAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int64_t inLocation, int32_t *ioNumSamples)=0
CALL: Randomly access audio from the timeline.
- virtual int32_t [GetSideChainInputNum](#) ()=0
CALL: Returns the index of the side chain input buffer.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.70.2 Member Function Documentation

14.70.2.1 GetAudio()

```
virtual AAX_Result AAX_IACFHostProcessorDelegate::GetAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    int64_t inLocation,
    int32_t * ioNumSamples ) [pure virtual]
```

CALL: Randomly access audio from the timeline.

Called from within [AAX_IHostProcessor::RenderAudio\(\)](#), this method fills a buffer of samples with randomly-accessed data from the current input processing region on the timeline, including any extra samples such as processing "handles".

Note

Plug-ins that use this feature must set [AAX_eProperty_UsesRandomAccess](#) to true

It is not possible to retrieve samples from outside of the current input processing region

Always check the return value of this method before using the randomly-accessed samples

Parameters

in	<i>inAudioIns</i>	Timeline audio buffer(s). This must be set to <i>inAudioIns</i> from AAX_IHostProcessor::RenderAudio()
in	<i>inAudioInCount</i>	Number of buffers in <i>inAudioIns</i> . This must be set to <i>inAudioInCount</i> from AAX_IHostProcessor::RenderAudio()
in	<i>inLocation</i>	A sample location relative to the beginning of the currently processed region, e.g. a value of 0 corresponds to the timeline location returned by AAX_CHostProcessor::GetSrcStart()
in, out	<i>ioNumSamples</i>	<ul style="list-style-type: none"> Input: The maximum number of samples to read. Output: The actual number of samples that were read from the timeline

14.70.2.2 GetSideChainInputNum()

```
virtual int32_t AAX_IACFHostProcessorDelegate::GetSideChainInputNum ( ) [pure virtual]
```

CALL: Returns the index of the side chain input buffer.

Called from within [AAX_IHostProcessor::RenderAudio\(\)](#), this method returns the index of the side chain input sample buffer within *inAudioIns*.

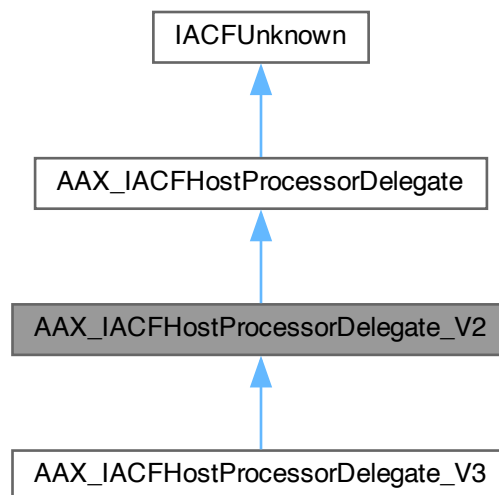
The documentation for this class was generated from the following file:

- [AAX_IACFHostProcessorDelegate.h](#)

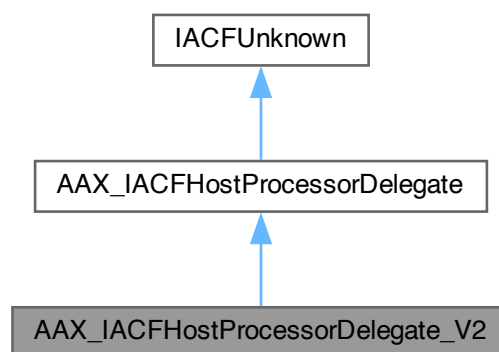
14.71 AAX_IACFHostProcessorDelegate_V2 Class Reference

```
#include <AAX_IACFHostProcessorDelegate.h>
```

Inheritance diagram for AAX_IACFHostProcessorDelegate_V2:



Collaboration diagram for AAX_IACFHostProcessorDelegate_V2:



14.71.1 Description

Versioned interface for host methods specific to offline processing.

Public Member Functions

- virtual [AAX_Result ForceAnalyze](#) ()=0
CALL: Request an analysis pass.

Public Member Functions inherited from [AAX_IACFHostProcessorDelegate](#)

- virtual [AAX_Result GetAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int64_t inLocation, int32_t *ioNumSamples)=0
CALL: Randomly access audio from the timeline.
- virtual int32_t [GetSideChainInputNum](#) ()=0
CALL: Returns the index of the side chain input buffer.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.71.2 Member Function Documentation

14.71.2.1 ForceAnalyze()

```
virtual AAX\_Result AAX_IACFHostProcessorDelegate_V2::ForceAnalyze ( ) [pure virtual]
```

CALL: Request an analysis pass.

Call this method to request an analysis pass from within the plug-in. Most plug-ins should rely on the host to trigger analysis passes when appropriate. However, plug-ins that require an analysis pass a) outside of the context of host-driven render or analysis, or b) when internal plug-in data changes may need to call [ForceAnalyze\(\)](#).

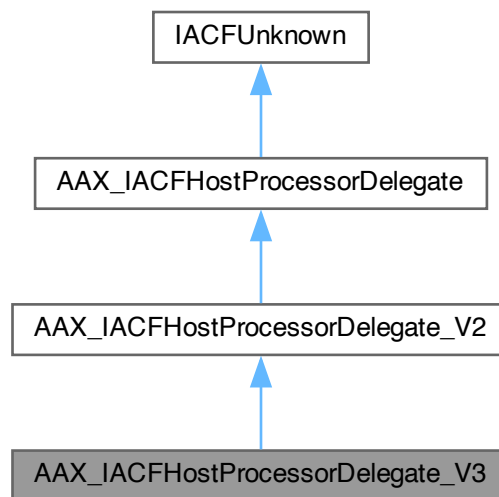
The documentation for this class was generated from the following file:

- [AAX_IACFHostProcessorDelegate.h](#)

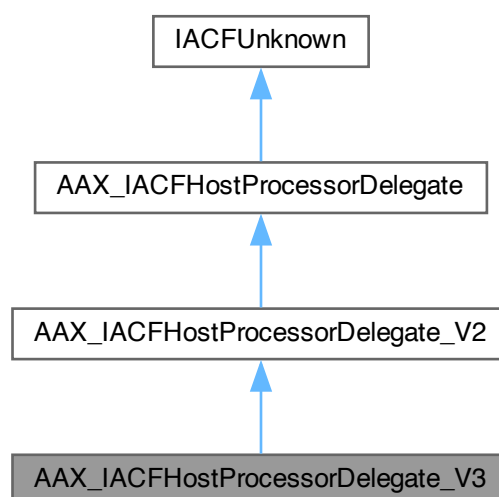
14.72 AAX_IACFHostProcessorDelegate_V3 Class Reference

```
#include <AAX_IACFHostProcessorDelegate.h>
```

Inheritance diagram for AAX_IACFHostProcessorDelegate_V3:



Collaboration diagram for AAX_IACFHostProcessorDelegate_V3:



14.72.1 Description

Versioned interface for host methods specific to offline processing.

Public Member Functions

- virtual [AAX_Result ForceProcess](#) ()=0
CALL: Request a process pass.

Public Member Functions inherited from [AAX_IACFHostProcessorDelegate_V2](#)

- virtual [AAX_Result ForceAnalyze](#) ()=0
CALL: Request an analysis pass.

Public Member Functions inherited from [AAX_IACFHostProcessorDelegate](#)

- virtual [AAX_Result GetAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int64_t inLocation, int32_t *ioNumSamples)=0
CALL: Randomly access audio from the timeline.
- virtual int32_t [GetSideChainInputNum](#) ()=0
CALL: Returns the index of the side chain input buffer.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.72.2 Member Function Documentation

14.72.2.1 ForceProcess()

```
virtual AAX\_Result AAX_IACFHostProcessorDelegate_V3::ForceProcess ( ) [pure virtual]
```

CALL: Request a process pass.

Call this method to request a process pass from within the plug-in. If [AAX_eProperty_RequiresAnalysis](#) is defined, the resulting process pass will be preceded by an analysis pass. This method should only be used in rare circumstances by plug-ins that must launch processing outside of the normal host AudioSuite workflow.

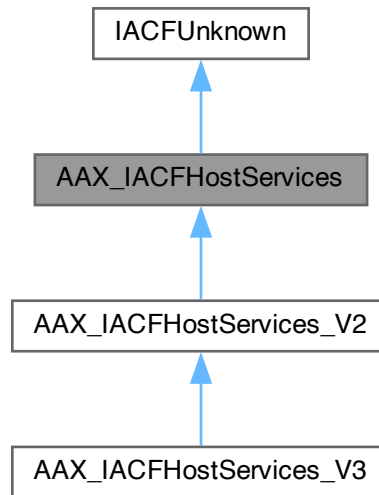
The documentation for this class was generated from the following file:

- [AAX_IACFHostProcessorDelegate.h](#)

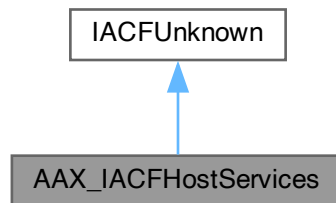
14.73 AAX_IACFHostServices Class Reference

```
#include <AAX_IACFHostServices.h>
```

Inheritance diagram for AAX_IACFHostServices:



Collaboration diagram for AAX_IACFHostServices:



14.73.1 Description

Versioned interface to diagnostic and debugging services provided by the AAX host.

Public Member Functions

- virtual [AAX_Result Assert](#) (const char *iFile, int32_t iLine, const char *iNote)=0
- virtual [AAX_Result Trace](#) (int32_t iPriority, const char *iMessage)=0

Log a trace message.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.73.2 Member Function Documentation

14.73.2.1 Assert()

```
virtual AAX\_Result AAX_IACFHostServices::Assert (
    const char * iFile,
    int32_t iLine,
    const char * iNote ) [pure virtual]
```

Deprecated Legacy version of [AAX_IACFHostServices_V3::HandleAssertFailure\(\)](#) implemented by older hosts

Prior to [AAX_IACFHostServices_V3::HandleAssertFailure\(\)](#), the [AAX_ASSERT](#) macro, a wrapper around [Assert\(\)](#), was only compiled into debug plug-in builds. [AAX_ASSERT](#) is now compiled in to all plug-in builds and the original debug-only form is available through [AAX_DEBUGASSERT](#).

Because the implementation of [Assert\(\)](#) in the host is not aware of the plug-in's build configuration, older hosts implemented this method with a warning dialog in all cases. Newer hosts - those which implement [HandleAssertFailure\(\)](#) - will log assertion failures but will not present any user dialog in shipping builds of the host software.

In order to prevent assertion failure dialogs from appearing to users who run new builds of plug-ins containing [AAX_ASSERT](#) calls in older hosts the deprecated [Assert\(\)](#) method should only be called from debug plug-in builds.

14.73.2.2 Trace()

```
virtual AAX\_Result AAX_IACFHostServices::Trace (
    int32_t iPriority,
    const char * iMessage ) [pure virtual]
```

Log a trace message.

Parameters

in	<i>iPriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iMessage</i>	Message string to log

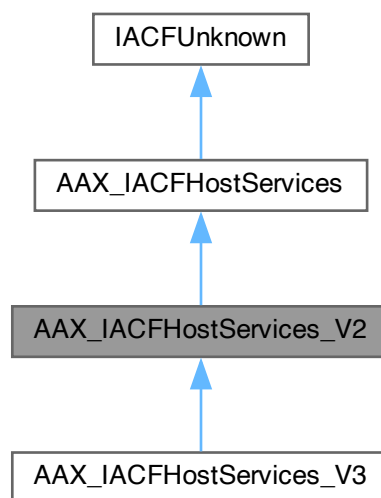
The documentation for this class was generated from the following file:

- [AAX_IACFHostServices.h](#)

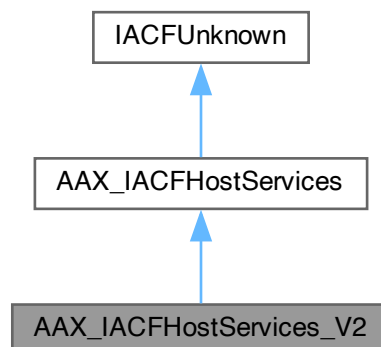
14.74 AAX_IACFHostServices_V2 Class Reference

```
#include <AAX_IACFHostServices.h>
```

Inheritance diagram for AAX_IACFHostServices_V2:



Collaboration diagram for AAX_IACFHostServices_V2:



14.74.1 Description

V2 of versioned interface to diagnostic and debugging services provided by the AAX host.

Public Member Functions

- virtual [AAX_Result StackTrace](#) (int32_t iTracePriority, int32_t iStackTracePriority, const char *iMessage)=0
Log a trace message or a stack trace.

Public Member Functions inherited from [AAX_IACFHostServices](#)

- virtual [AAX_Result Assert](#) (const char *iFile, int32_t iLine, const char *iNote)=0
- virtual [AAX_Result Trace](#) (int32_t iPriority, const char *iMessage)=0
Log a trace message.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.74.2 Member Function Documentation

14.74.2.1 StackTrace()

```
virtual AAX\_Result AAX_IACFHostServices_V2::StackTrace (
    int32_t iTracePriority,
    int32_t iStackTracePriority,
    const char * iMessage ) [pure virtual]
```

Log a trace message or a stack trace.

If the logging output filtering is set to include logs with `iStackTracePriority` then both the logging message and a stack trace will be emitted, regardless of `iTracePriority`.

If the logging output filtering is set to include logs with `iTracePriority` but to exclude logs with `iStackTracePriority` then this will emit a normal log with no stack trace.

Parameters

in	<i>iTracePriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iStackTracePriority</i>	Priority of the stack trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
Generated by Doxygen		
in	<i>iMessage</i>	Message string to log

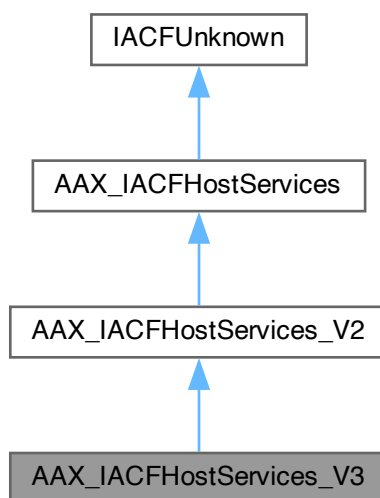
The documentation for this class was generated from the following file:

- [AAX_IACFHostServices.h](#)

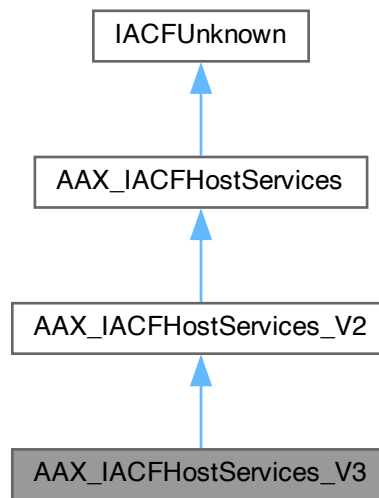
14.75 AAX_IACFHostServices_V3 Class Reference

```
#include <AAX_IACFHostServices.h>
```

Inheritance diagram for AAX_IACFHostServices_V3:



Collaboration diagram for AAX_IACFHostServices_V3:



14.75.1 Description

V3 of versioned interface to diagnostic and debugging services provided by the AAX host.

Public Member Functions

- virtual [AAX_Result HandleAssertFailure](#) (const char *iFile, int32_t iLine, const char *iNote, int32_t iFlags) const =0
Handle an assertion failure.

Public Member Functions inherited from [AAX_IACFHostServices_V2](#)

- virtual [AAX_Result StackTrace](#) (int32_t iTracePriority, int32_t iStackTracePriority, const char *iMessage)=0
Log a trace message or a stack trace.

Public Member Functions inherited from [AAX_IACFHostServices](#)

- virtual [AAX_Result Assert](#) (const char *iFile, int32_t iLine, const char *iNote)=0
- virtual [AAX_Result Trace](#) (int32_t iPriority, const char *iMessage)=0
Log a trace message.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.75.2 Member Function Documentation

14.75.2.1 HandleAssertFailure()

```
virtual AAX\_Result AAX_IACFHostServices_V3::HandleAssertFailure (
    const char * iFile,
    int32_t iLine,
    const char * iNote,
    int32_t iFlags ) const [pure virtual]
```

Handle an assertion failure.

Use this method to delegate assertion failure handling to the host

Use *iFlags* to request that specific behavior be included when handling the failure. This request may not be fulfilled by the host, and absence of a flag does not preclude the host from using that behavior when handling the failure.

Parameters

in	<i>iFile</i>	The name of the file containing the assert check. Usually <code>__FILE__</code>
in	<i>iLine</i>	The line number of the assert check. Usually <code>__LINE__</code>
in	<i>iNote</i>	Text to display related to the assert. Usually the condition which failed
in	<i>iFlags</i>	Bitfield of AAX_EAssertFlags to request specific handling behavior

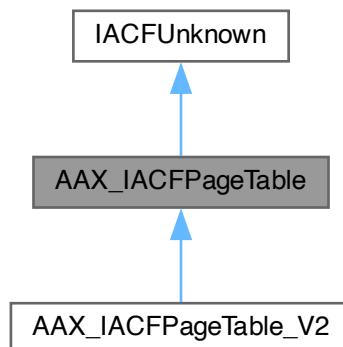
The documentation for this class was generated from the following file:

- [AAX_IACFHostServices.h](#)

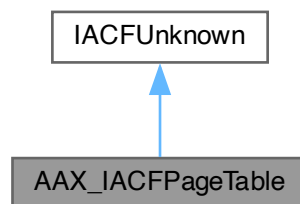
14.76 AAX_IACFPageTable Class Reference

```
#include <AAX_IACFPageTable.h>
```

Inheritance diagram for AAX_IACFPageTable:



Collaboration diagram for AAX_IACFPageTable:



14.76.1 Description

Versioned interface to the host's representation of a plug-in instance's page table.

Public Member Functions

- virtual [AAX_Result Clear](#) ()=0
Clears all parameter mappings from the table.
- virtual [AAX_Result Empty](#) (AAX_CBoolean &oEmpty) const =0
Indicates whether the table is empty.
- virtual [AAX_Result GetNumPages](#) (int32_t &oNumPages) const =0
Get the number of pages currently in this table.
- virtual [AAX_Result InsertPage](#) (int32_t iPage)=0
Insert a new empty page before the page at index iPage.

- virtual [AAX_Result RemovePage](#) (int32_t iPage)=0
Remove the page at index iPage.
- virtual [AAX_Result GetNumMappedParameterIDs](#) (int32_t iPage, int32_t &oNumParameterIdentifiers) const =0
Returns the total number of parameter IDs which are mapped to a page.
- virtual [AAX_Result ClearMappedParameter](#) (int32_t iPage, int32_t iIndex)=0
Clear the parameter at a particular index in this table.
- virtual [AAX_Result GetMappedParameterID](#) (int32_t iPage, int32_t iIndex, [AAX_IString](#) &oParameterIdentifier) const =0
Get the parameter identifier which is currently mapped to an index in this table.
- virtual [AAX_Result MapParameterID](#) ([AAX_CParamID](#) iParameterIdentifier, int32_t iPage, int32_t iIndex)=0
Map a parameter to this table.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32 ACFMETHODCALLTYPE AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32 ACFMETHODCALLTYPE Release](#) (void)=0
Decrements reference count.

14.76.2 Member Function Documentation

14.76.2.1 Clear()

```
virtual AAX\_Result AAX_IACFPageTable::Clear ( ) [pure virtual]
```

Clears all parameter mappings from the table.

This method does not clear any parameter name variations from the table. For that, use [AAX_IPageTable::ClearParameterNameVariations](#) or [AAX_IPageTable::ClearNameVariationsForParameter\(\)](#)

14.76.2.2 Empty()

```
virtual AAX\_Result AAX_IACFPageTable::Empty (
    AAX\_CBoolean & oEmpty ) const [pure virtual]
```

Indicates whether the table is empty.

A table is empty if it contains no pages. A table which contains pages but no parameter assignments is not empty. A table which has associated parameter name variations but no pages is empty.

Parameters

out	<i>oEmpty</i>	true if this table is empty
-----	---------------	-----------------------------

14.76.2.3 GetNumPages()

```
virtual AAX_Result AAX_IACFPageTable::GetNumPages (
    int32_t & oNumPages ) const [pure virtual]
```

Get the number of pages currently in this table.

Parameters

out	<i>oNumPages</i>	The number of pages which are present in the page table. Some pages might not contain any parameter assignments.
-----	------------------	--

14.76.2.4 InsertPage()

```
virtual AAX_Result AAX_IACFPageTable::InsertPage (
    int32_t iPage ) [pure virtual]
```

Insert a new empty page before the page at index *iPage*.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the total number of pages

Parameters

in	<i>iPage</i>	The insertion point page index
----	--------------	--------------------------------

14.76.2.5 RemovePage()

```
virtual AAX_Result AAX_IACFPageTable::RemovePage (
    int32_t iPage ) [pure virtual]
```

Remove the page at index *iPage*.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

Parameters

in	<i>iPage</i>	The target page index
----	--------------	-----------------------

14.76.2.6 GetNumMappedParameterIDs()

```
virtual AAX_Result AAX_IACFPageTable::GetNumMappedParameterIDs (
    int32_t iPage,
    int32_t & oNumParameterIdentifiers ) const [pure virtual]
```

Returns the total number of parameter IDs which are mapped to a page.

Note

The number of mapped parameter IDs does not correspond to the actual slot indices of the parameter assignments. For example, a page could have three total parameter assignments with parameters mapped to slots 2, 4, and 6.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

Parameters

in	<i>iPage</i>	The target page index
out	<i>oNumParameterIdentifiers</i>	The number of parameter identifiers which are mapped to the target page

14.76.2.7 ClearMappedParameter()

```
virtual AAX_Result AAX_IACFPageTable::ClearMappedParameter (
    int32_t iPage,
    int32_t iIndex ) [pure virtual]
```

Clear the parameter at a particular index in this table.

Returns

[AAX_SUCCESS](#) even if no parameter was mapped at the given index (the index is still clear)

Parameters

in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page

14.76.2.8 GetMappedParameterID()

```
virtual AAX_Result AAX_IACFPageTable::GetMappedParameterID (
    int32_t iPage,
    int32_t iIndex,
    AAX_IString & oParameterIdentifier ) const [pure virtual]
```

Get the parameter identifier which is currently mapped to an index in this table.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if no parameter is mapped at the specified page/index location

Parameters

in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page
out	<i>oParameterIdentifier</i>	The identifier used for the mapped parameter in the page table (may be parameter name or ID)

14.76.2.9 MapParameterID()

```
virtual AAX_Result AAX_IACFPageTable::MapParameterID (
    AAX_CParamID iParameterIdentifier,
    int32_t iPage,
    int32_t iIndex ) [pure virtual]
```

Map a parameter to this table.

If *iParameterIdentifier* is an empty string then the parameter assignment will be cleared

Returns

[AAX_ERROR_NULL_ARGUMENT](#) if *iParameterIdentifier* is null

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

[AAX_ERROR_INVALID_ARGUMENT](#) if *iIndex* is negative

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter which will be mapped
in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page

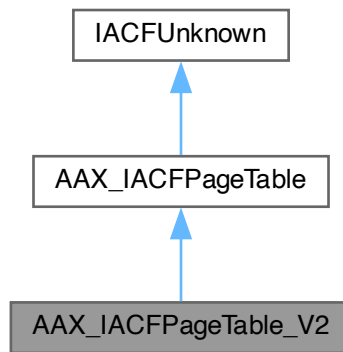
The documentation for this class was generated from the following file:

- [AAX_IACFPageTable.h](#)

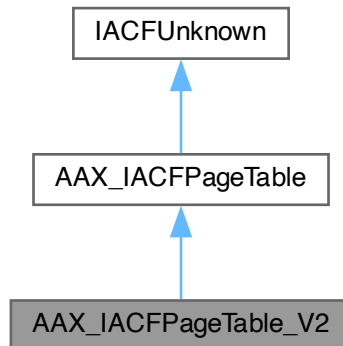
14.77 AAX_IACFPageTable_V2 Class Reference

```
#include <AAX_IACFPageTable.h>
```

Inheritance diagram for AAX_IACFPageTable_V2:



Collaboration diagram for AAX_IACFPageTable_V2:



14.77.1 Description

Versioned interface to the host's representation of a plug-in instance's page table.

Public Member Functions

- virtual [AAX_Result GetNumParametersWithNameVariations](#) (int32_t &oNumParameterIdentifiers) const =0
- virtual [AAX_Result GetNameVariationParameterIDAtIndex](#) (int32_t iIndex, [AAX_IString](#) &oParameterIdentifier) const =0
- virtual [AAX_Result GetNumNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t &oNumVariations) const =0
- virtual [AAX_Result GetParameterNameVariationAtIndex](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t iIndex, [AAX_IString](#) &oNameVariation, int32_t &oLength) const =0
- virtual [AAX_Result GetParameterNameVariationOfLength](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t iLength, [AAX_IString](#) &oNameVariation) const =0
- virtual [AAX_Result ClearParameterNameVariations](#) ()=0
- virtual [AAX_Result ClearNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier)=0
- virtual [AAX_Result SetParameterNameVariation](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, const [AAX_IString](#) &iNameVariation, int32_t iLength)=0

Public Member Functions inherited from [AAX_IACFPageTable](#)

- virtual [AAX_Result Clear](#) ()=0
Clears all parameter mappings from the table.
- virtual [AAX_Result Empty](#) ([AAX_CBoolean](#) &oEmpty) const =0
Indicates whether the table is empty.
- virtual [AAX_Result GetNumPages](#) (int32_t &oNumPages) const =0
Get the number of pages currently in this table.
- virtual [AAX_Result InsertPage](#) (int32_t iPage)=0
Insert a new empty page before the page at index iPage.
- virtual [AAX_Result RemovePage](#) (int32_t iPage)=0
Remove the page at index iPage.
- virtual [AAX_Result GetNumMappedParameterIDs](#) (int32_t iPage, int32_t &oNumParameterIdentifiers) const =0
Returns the total number of parameter IDs which are mapped to a page.
- virtual [AAX_Result ClearMappedParameter](#) (int32_t iPage, int32_t iIndex)=0
Clear the parameter at a particular index in this table.
- virtual [AAX_Result GetMappedParameterID](#) (int32_t iPage, int32_t iIndex, [AAX_IString](#) &oParameterIdentifier) const =0
Get the parameter identifier which is currently mapped to an index in this table.
- virtual [AAX_Result MapParameterID](#) ([AAX_CParamID](#) iParameterIdentifier, int32_t iPage, int32_t iIndex)=0
Map a parameter to this table.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32 ACFMETHODCALLTYPE AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32 ACFMETHODCALLTYPE Release](#) (void)=0
Decrements reference count.

14.77.2 Member Function Documentation

14.77.2.1 GetNumParametersWithNameVariations()

```
virtual AAX_Result AAX_IACFPageTable_V2::GetNumParametersWithNameVariations (
    int32_t & oNumParameterIdentifiers ) const [pure virtual]
```

Get the number of parameters with name variations defined for the current table type

Provides the number of parameters with `lt;ControlNameVariationslt;` which are explicitly defined for the current page table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNameVariationParameterIDAtIndex\(\)](#)

Parameters

out	<i>oNumParameterIdentifiers</i>	The number of parameters with name variations explicitly associated with the current table type.
-----	---------------------------------	--

14.77.2.2 GetNameVariationParameterIDAtIndex()

```
virtual AAX_Result AAX_IACFPageTable_V2::GetNameVariationParameterIDAtIndex (
    int32_t iIndex,
    AAX_IString & oParameterIdentifier ) const [pure virtual]
```

Get the identifier for a parameter with name variations defined for the current table type

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNumParametersWithNameVariations\(\)](#)

Parameters

in	<i>iIndex</i>	The target parameter index within the list of parameters with explicit name variations defined for this table type.
out	<i>oParameterIdentifier</i>	The identifier used for the parameter in the page table name variations list (may be parameter name or ID)

14.77.2.3 GetNumNameVariationsForParameter()

```
virtual AAX_Result AAX_IACFPageTable_V2::GetNumNameVariationsForParameter (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t & oNumVariations ) const [pure virtual]
```

Get the number of name variations defined for a parameter

Provides the number of `lt;ControlNameVariationslt;` which are explicitly defined for `iParameterIdentifier` for the current page table type. No fallback logic is used to resolve this to the list of variations which would actually be used for an attached control surface if no explicit variations are defined for the current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetParameterNameVariationAtIndex\(\)](#)

Returns

[AAX_SUCCESS](#) and provides zero to `oNumVariations` if `iParameterIdentifier` is not found

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
out	<i>oNumVariations</i>	The number of name variations which are defined for this parameter and explicitly associated with the current table type.

14.77.2.4 GetParameterNameVariationAtIndex()

```
virtual AAX_Result AAX_IACFPageTable_V2::GetParameterNameVariationAtIndex (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t iIndex,
    AAX_IString & oNameVariation,
    int32_t & oLength ) const [pure virtual]
```

Get a parameter name variation from the page table

Only returns `lt;ControlNameVariationslt;` which are explicitly defined for the current page table type. No fallback logic is used to resolve this to the abbreviation which would actually be shown on an attached control surface if no explicit variation is defined for the current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNumNameVariationsForParameter\(\)](#)

See also

- [AAX_IPageTable::GetParameterNameVariationOfLength\(\)](#)

Returns

[AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME](#) if no suitable variation is defined for this table

[AAX_ERROR_ARGUMENT_OUT_OF_RANGE](#) if `iIndex` is out of range

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iIndex</i>	Index of the name variation
out	<i>oNameVariation</i>	The name variation, if one is explicitly defined for this table type
out	<i>oLength</i>	The length value for this name variation. This corresponds to the variation's <code>sz</code> attribute in the page table XML and may be different from the string length of <code>iNameVariation</code> .

14.77.2.5 GetParameterNameVariationOfLength()

```
virtual AAX_Result AAX_IACFPPageTable_V2::GetParameterNameVariationOfLength (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t iLength,
    AAX_IString & oNameVariation ) const [pure virtual]
```

Get a parameter name variation of a particular length from the page table

Only returns `lt;ControlNameVariationslt;` which are explicitly defined of `iLength` for the current page table type. No fallback logic is used to resolve this to the abbreviation which would actually be shown on an attached control surface if no explicit variation is defined for the specified length or current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetParameterNameVariationAtIndex\(\)](#)

Returns

[AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME](#) if no suitable variation is defined for this table

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iLength</i>	The variation length to check, i.e. the <code>sz</code> attribute for the name variation in the page table XML
out	<i>oNameVariation</i>	The name variation, if one is explicitly defined for this table type and <code>iLength</code>

14.77.2.6 ClearParameterNameVariations()

```
virtual AAX\_Result AAX_IACFPageTable_V2::ClearParameterNameVariations ( ) [pure virtual]
```

Clears all name variations for the current page table type

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

[AAX_IPageTable::Clear\(\)](#)

[AAX_IPageTable::ClearNameVariationsForParameter\(\)](#)

14.77.2.7 ClearNameVariationsForParameter()

```
virtual AAX\_Result AAX_IACFPageTable_V2::ClearNameVariationsForParameter (
    AAX\_CPageTableParamID iParameterIdentifier ) [pure virtual]
```

Clears all name variations for a single parameter for the current page table type

Warning

This will invalidate the list of parameter name variations indices, i.e. the parameter identifier associated with each index by [AAX_IPageTable::GetNameVariationParameterIDAtIndex\(\)](#)

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

[AAX_IPageTable::Clear\(\)](#)

[AAX_IPageTable::ClearParameterNameVariations\(\)](#)

Returns

[AAX_SUCCESS](#) and provides zero to `oNumVariations` if `iParameterIdentifier` is not found

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
----	-----------------------------	----------------------------------

14.77.2.8 SetParameterNameVariation()

```
virtual AAX_Result AAX_IACFPageTable_V2::SetParameterNameVariation (
    AAX_CPageTableParamID iParameterIdentifier,
    const AAX_IString & iNameVariation,
    int32_t iLength ) [pure virtual]
```

Sets a name variation explicitly for the current page table type

This will add a new name variation or overwrite the existing name variation with the same length which is defined for the current table type.

Warning

If no name variation previously existed for this parameter then this will invalidate the list of parameter name variations indices, i.e. the parameter identifier associated with each index by [AAX_IPageTable::GetNameVariationParameterIDAt](#)

Note

Normally parameter name variations are only used with the 'PgTL' table type

Returns

AAX_ERROR_INVALID_ARGUMENT if *iNameVariation* is empty or if *iLength* is less than zero

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iNameVariation</i>	The new parameter name variation
in	<i>iLength</i>	The length value for this name variation. This corresponds to the variation's <i>sz</i> attribute in the page table XML and is not required to match the length of <i>iNameVariation</i> .

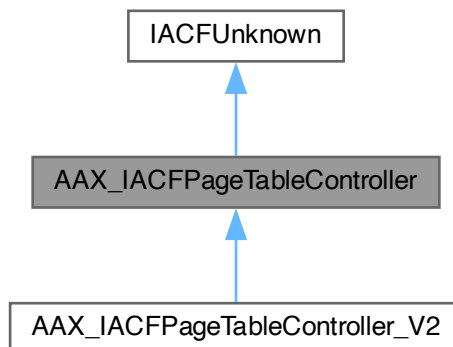
The documentation for this class was generated from the following file:

- [AAX_IACFPageTable.h](#)

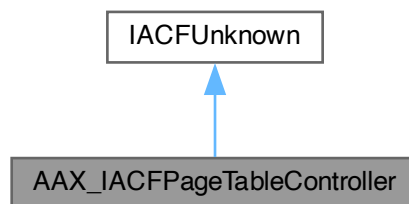
14.78 AAX_IACFPageTableController Class Reference

```
#include <AAX_IACFPageTableController.h>
```

Inheritance diagram for AAX_IACFPageTableController:



Collaboration diagram for AAX_IACFPageTableController:



14.78.1 Description

Interface for host operations related to the page tables for this plug-in.

Note

In the AAX Library, access to this interface is provided through [AAX_IController](#)

Public Member Functions

- virtual [AAX_Result CopyTableForEffect](#) ([AAX_CPropertyValue](#) inManufacturerID, [AAX_CPropertyValue](#) inProductID, [AAX_CPropertyValue](#) inPlugInID, [uint32_t](#) inTableType, [int32_t](#) inTablePageSize, [IACFUnknown](#) *oPageTable) const =0
- virtual [AAX_Result CopyTableOfLayoutForEffect](#) (const char *inEffectID, const char *inLayoutName, [uint32_t](#) inTableType, [int32_t](#) inTablePageSize, [IACFUnknown](#) *oPageTable) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.78.2 Member Function Documentation

14.78.2.1 CopyTableForEffect()

```
virtual AAX\_Result AAX_IACFPageTableController::CopyTableForEffect (
    AAX\_CPropertyValue inManufacturerID,
    AAX\_CPropertyValue inProductID,
    AAX\_CPropertyValue inPlugInID,
    uint32_t inTableType,
    int32_t inTablePageSize,
    IACFUnknown * oPageTable ) const [pure virtual]
```

Copy the current page table data for a particular plug-in type.

The host will reject the copy and return an error if the requested plug-in type is unknown, if `inTableType` is unknown or if `inTablePageSize` is not a supported size for the given table type.

The host may also restrict plug-ins to only copying page table data from certain plug-in types, such as plug-ins from the same manufacturer or plug-in types within the same effect.

See [Page Table Guide](#) for more information about page tables.

Returns

[AAX_ERROR_NULL_ARGUMENT](#) if `oPageTable` is null

[AAX_ERROR_INVALID_ARGUMENT](#) if no valid page table mapping can be created due to the specified arguments

Parameters

in	<i>inManufacturerID</i>	Manufacturer ID of the desired plug-in type
in	<i>inProductID</i>	Product ID of the desired plug-in type
in	<i>inPlugInID</i>	Type ID of the desired plug-in type (AAX_eProperty_PlugInID_Native , AAX_eProperty_PlugInID_TI)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.
out	<i>oPageTable</i>	The page table object to which the page table data should be copied. <code>oPageTable</code> must support AAX_IACFPageTable

See also

[AAX_IController::CreateTableCopyForEffect\(\)](#)

14.78.2.2 CopyTableOfLayoutForEffect()

```
virtual AAX\_Result AAX_IACFPageTableController::CopyTableOfLayoutForEffect (
    const char * inEffectID,
    const char * inLayoutName,
    uint32_t inTableType,
    int32_t inTablePageSize,
    IACFUnknown * oPageTable ) const [pure virtual]
```

Copy the current page table data for a particular plug-in effect and page table layout.

The host will reject the copy and return an error if the requested effect ID is unknown or if *inLayoutName* is not a valid layout name for the page tables registered for the effect.

The host may also restrict plug-ins to only copying page table data from certain effects, such as effects registered within the current [AAX](#) plug-in bundle.

See [Page Table Guide](#) for more information about page tables.

Returns

[AAX_ERROR_NULL_ARGUMENT](#) if *inEffectID*, *inLayoutName*, or *oPageTable* is null

[AAX_ERROR_INVALID_ARGUMENT](#) if no valid page table mapping can be created due to the specified arguments

Parameters

in	<i>inEffectID</i>	Effect ID for the desired effect. See AAX_ICollection::AddEffect()
in	<i>inLayoutName</i>	Page table layout name ("name" attribute of the <code>PTLayout</code> XML tag)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.
out	<i>oPageTable</i>	The page table object to which the page table data should be copied. <i>oPageTable</i> must support AAX_IACFPageTable

See also

[AAX_IController::CreateTableCopyForLayout\(\)](#)

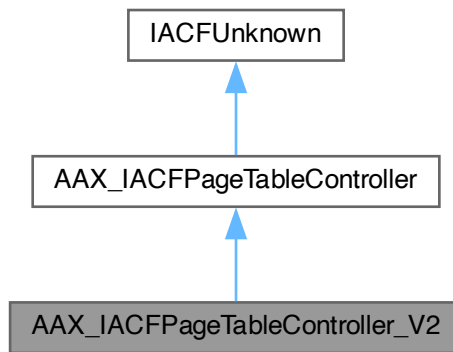
The documentation for this class was generated from the following file:

- [AAX_IACFPageTableController.h](#)

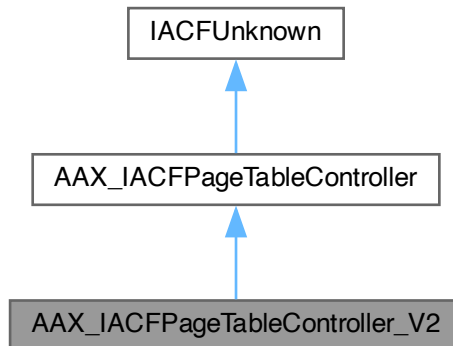
14.79 AAX_IACFPageTableController_V2 Class Reference

```
#include <AAX_IACFPageTableController.h>
```

Inheritance diagram for AAX_IACFPageTableController_V2:



Collaboration diagram for AAX_IACFPageTableController_V2:



14.79.1 Description

Interface for host operations related to the page tables for this plug-in.

Note

In the AAX Library, access to this interface is provided through [AAX_IController](#)

Public Member Functions

- virtual [AAX_Result CopyTableForEffectFromFile](#) (const char *inPageTableFilePath, [AAX_ETextEncoding](#) inFilePathEncoding, [AAX_CPropertyValue](#) inManufacturerID, [AAX_CPropertyValue](#) inProductID, [AAX_CPropertyValue](#) inPlugInID, uint32_t inTableType, int32_t inTablePageSize, [IACFUnknown](#) *oPageTable) const =0
- virtual [AAX_Result CopyTableOfLayoutFromFile](#) (const char *inPageTableFilePath, [AAX_ETextEncoding](#) inFilePathEncoding, const char *inLayoutName, uint32_t inTableType, int32_t inTablePageSize, [IACFUnknown](#) *oPageTable) const =0

Public Member Functions inherited from [AAX_IACFPageTableController](#)

- virtual [AAX_Result CopyTableForEffect](#) ([AAX_CPropertyValue](#) inManufacturerID, [AAX_CPropertyValue](#) inProductID, [AAX_CPropertyValue](#) inPlugInID, uint32_t inTableType, int32_t inTablePageSize, [IACFUnknown](#) *oPageTable) const =0
- virtual [AAX_Result CopyTableOfLayoutForEffect](#) (const char *inEffectID, const char *inLayoutName, uint32_t inTableType, int32_t inTablePageSize, [IACFUnknown](#) *oPageTable) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.79.2 Member Function Documentation

14.79.2.1 CopyTableForEffectFromFile()

```
virtual AAX\_Result AAX_IACFPageTableController_V2::CopyTableForEffectFromFile (
    const char * inPageTableFilePath,
    AAX\_ETextEncoding inFilePathEncoding,
    AAX\_CPropertyValue inManufacturerID,
    AAX\_CPropertyValue inProductID,
    AAX\_CPropertyValue inPlugInID,
    uint32_t inTableType,
    int32_t inTablePageSize,
    IACFUnknown * oPageTable ) const [pure virtual]
```

Returns

[AAX_ERROR_NULL_ARGUMENT](#) if inPageTableFilePath or oPageTable is null

[AAX_ERROR_UNSUPPORTED_ENCODING](#) if inFilePathEncoding has unsupported encoding value

[AAX_ERROR_INVALID_ARGUMENT](#) if no valid page table mapping can be created due to the specified arguments

Parameters

in	<i>inPageTableFilePath</i>	Path to XML page table file.
in	<i>inFilePathEncoding</i>	File path text encoding.
in	<i>inManufacturerID</i>	Manufacturer ID of the desired plug-in type
in	<i>inProductID</i>	Product ID of the desired plug-in type
in	<i>inPlugInID</i>	Type ID of the desired plug-in type (AAX_eProperty_PlugInID_Native , AAX_eProperty_PlugInID_TI)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.
out	<i>oPageTable</i>	The page table object to which the page table data should be copied. <i>oPageTable</i> must support AAX_IACFPageTable

See also

[AAX_IController::CreateTableCopyForEffect\(\)](#)

14.79.2.2 CopyTableOfLayoutFromFile()

```
virtual AAX_Result AAX_IACFPageTableController_V2::CopyTableOfLayoutFromFile (
    const char * inPageTableFilePath,
    AAX_ETextEncoding inFilePathEncoding,
    const char * inLayoutName,
    uint32_t inTableType,
    int32_t inTablePageSize,
    IACFUnknown * oPageTable ) const [pure virtual]
```

Returns

[AAX_ERROR_NULL_ARGUMENT](#) if *inPageTableFilePath*, *inLayoutName*, or *oPageTable* is null

[AAX_ERROR_UNSUPPORTED_ENCODING](#) if *inFilePathEncoding* has unsupported encoding value

[AAX_ERROR_INVALID_ARGUMENT](#) if no valid page table mapping can be created due to the specified arguments

Parameters

in	<i>inPageTableFilePath</i>	Path to XML page table file.
in	<i>inFilePathEncoding</i>	File path text encoding.
in	<i>inLayoutName</i>	Page table layout name ("name" attribute of the PTLayout XML tag)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.
out	<i>oPageTable</i>	The page table object to which the page table data should be copied. <i>oPageTable</i> must support AAX_IACFPageTable

See also

[AAX_IController::CreateTableCopyForLayout\(\)](#)

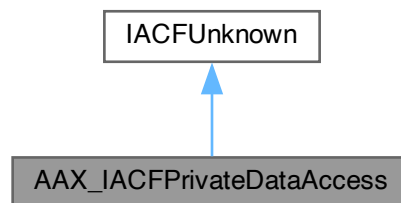
The documentation for this class was generated from the following file:

- [AAX_IACFPageTableController.h](#)

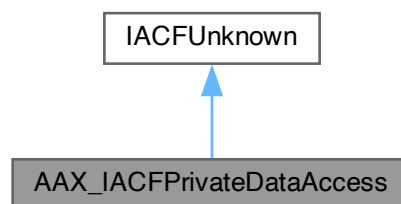
14.80 AAX_IACFPrivateDataAccess Class Reference

```
#include <AAX_IACFPrivateDataAccess.h>
```

Inheritance diagram for AAX_IACFPrivateDataAccess:



Collaboration diagram for AAX_IACFPrivateDataAccess:



14.80.1 Description

Interface for the AAX host's data access functionality.

Public Member Functions

- virtual [AAX_Result ReadPortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, void *outBuffer)=0
Read data directly from DSP at the given port.
- virtual [AAX_Result WritePortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, const void *inBuffer)=0
Write data directly to DSP at the given port.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.80.2 Member Function Documentation

14.80.2.1 ReadPortDirect()

```
virtual AAX\_Result AAX_IACFPrivateDataAccess::ReadPortDirect (
    AAX\_CFieldIndex inFieldIndex,
    const uint32_t inOffset,
    const uint32_t inSize,
    void * outBuffer ) [pure virtual]
```

Read data directly from DSP at the given port.

Note

Blocking

Parameters

in	<i>inFieldIndex</i>	The port to read from.
in	<i>inOffset</i>	Offset into data to start reading.
in	<i>inSize</i>	Amount of data to read (in bytes).
out	<i>outBuffer</i>	Pointer to storage for data to be read into.

14.80.2.2 WritePortDirect()

```
virtual AAX_Result AAX_IACFPrivateDataAccess::WritePortDirect (
    AAX_CFieldIndex inFieldIndex,
    const uint32_t inOffset,
    const uint32_t inSize,
    const void * inBuffer ) [pure virtual]
```

Write data directly to DSP at the given port.

Note

Blocking

Parameters

in	<i>inFieldIndex</i>	The port to write to.
in	<i>inOffset</i>	Offset into data to begin writing.
in	<i>inSize</i>	Amount of data to write (in bytes).
in	<i>inBuffer</i>	Pointer to data being written.

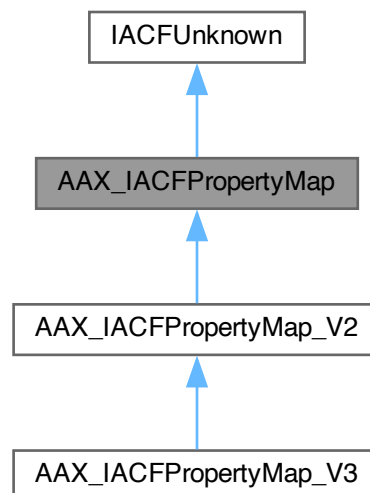
The documentation for this class was generated from the following file:

- [AAX_IACFPrivateDataAccess.h](#)

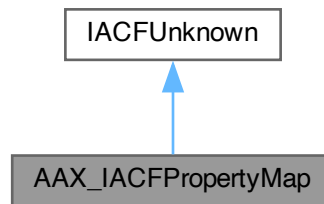
14.81 AAX_IACFPropertyMap Class Reference

```
#include <AAX_IACFPropertyMap.h>
```

Inheritance diagram for AAX_IACFPropertyMap:



Collaboration diagram for AAX_IACFPropertyMap:



14.81.1 Description

Versioned interface for an [AAX_IPropertyMap](#).

Public Member Functions

- virtual [AAX_CBoolean](#) `GetProperty` ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) *outValue) const =0
Get a property value from a property map.
- virtual [AAX_Result](#) `AddProperty` ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) inValue)=0
Add a property to a property map.
- virtual [AAX_Result](#) `RemoveProperty` ([AAX_EProperty](#) inProperty)=0
Remove a property from a property map.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.81.2 Member Function Documentation

14.81.2.1 GetProperty()

```
virtual AAX\_CBoolean AAX_IACFPropertyMap::GetProperty (
    AAX\_EProperty inProperty,
    AAX\_CPropertyValue * outValue ) const [pure virtual]
```

Get a property value from a property map.

Returns true if the selected property is supported, false if it is not

Parameters

in	<i>inProperty</i>	The property ID
out	<i>outValue</i>	The property value

14.81.2.2 AddProperty()

```
virtual AAX_Result AAX_IACFPropertyMap::AddProperty (
    AAX_EProperty inProperty,
    AAX_CPropertyValue inValue ) [pure virtual]
```

Add a property to a property map.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

14.81.2.3 RemoveProperty()

```
virtual AAX_Result AAX_IACFPropertyMap::RemoveProperty (
    AAX_EProperty inProperty ) [pure virtual]
```

Remove a property from a property map.

Parameters

in	<i>inProperty</i>	The property ID.
----	-------------------	------------------

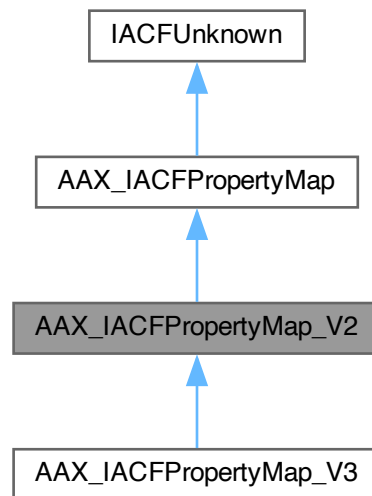
The documentation for this class was generated from the following file:

- [AAX_IACFPropertyMap.h](#)

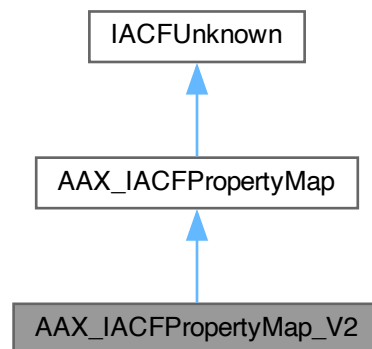
14.82 AAX_IACFPropertyMap_V2 Class Reference

```
#include <AAX_IACFPropertyMap.h>
```

Inheritance diagram for AAX_IACFPropertyMap_V2:



Collaboration diagram for AAX_IACFPropertyMap_V2:



14.82.1 Description

Versioned interface for an [AAX_IPropertyMap](#).

Public Member Functions

- virtual [AAX_Result](#) [AddPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPluginIdentifierTriad](#) *inPluginIDs, uint32_t inNumPluginIDs)=0

Add an array of plug-in IDs to a property map.

- virtual [AAX_CBoolean](#) [GetPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPlugInIdentifierTriad](#) **outPluginIDs, uint32_t *outNumPluginIDs) const =0

Get an array of plug-in IDs from a property map.

Public Member Functions inherited from [AAX_IACFPropertyMap](#)

- virtual [AAX_CBoolean](#) [GetProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) *outValue) const =0

Get a property value from a property map.

- virtual [AAX_Result](#) [AddProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) inValue)=0

Add a property to a property map.

- virtual [AAX_Result](#) [RemoveProperty](#) ([AAX_EProperty](#) inProperty)=0

Remove a property from a property map.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0

Returns pointers to supported interfaces.

- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0

Increments reference count.

- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0

Decrements reference count.

14.82.2 Member Function Documentation

14.82.2.1 AddPropertyWithIDArray()

```
virtual AAX\_Result AAX_IACFPropertyMap_V2::AddPropertyWithIDArray (
    AAX\_EProperty inProperty,
    const AAX\_SPlugInIdentifierTriad * inPluginIDs,
    uint32_t inNumPluginIDs ) [pure virtual]
```

Add an array of plug-in IDs to a property map.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inPluginIDs</i>	An array of AAX_SPlugInIdentifierTriad
in	<i>inNumPluginIDs</i>	The length of iPluginIDs

14.82.2.2 GetPropertyWithIDArray()

```
virtual AAX\_CBoolean AAX_IACFPropertyMap_V2::GetPropertyWithIDArray (
```

```

AAX_EProperty inProperty,
const AAX_SPlugInIdentifierTriad ** outPluginIDs,
uint32_t * outNumPluginIDs ) const [pure virtual]

```

Get an array of plug-in IDs from a property map.

Parameters

in	<i>inProperty</i>	The property ID.
out	<i>outPluginIDs</i>	A pointer that will be set to reference an array of AAX_SPlugInIdentifierTriad
in	<i>outNumPluginIDs</i>	The length of oPluginIDs

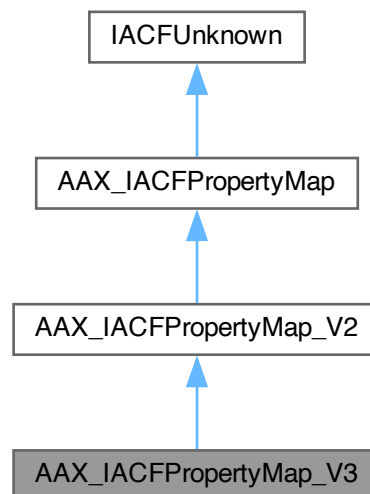
The documentation for this class was generated from the following file:

- [AAX_IACFPropertyMap.h](#)

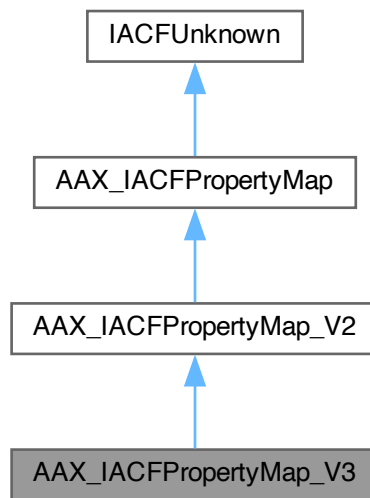
14.83 AAX_IACFPropertyMap_V3 Class Reference

```
#include <AAX_IACFPropertyMap.h>
```

Inheritance diagram for AAX_IACFPropertyMap_V3:



Collaboration diagram for AAX_IACFPropertyMap_V3:



14.83.1 Description

Versioned interface for an [AAX_IPropertyMap](#).

Public Member Functions

- virtual [AAX_CBoolean](#) [GetProperty64](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue64](#) *outValue) const =0
Get a property value from a property map.
- virtual [AAX_Result](#) [AddProperty64](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue64](#) inValue)=0
Add a property to a property map.

Public Member Functions inherited from [AAX_IACFPropertyMap_V2](#)

- virtual [AAX_Result](#) [AddPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPlugInIdentifierTriad](#) *inPluginIDs, uint32_t inNumPluginIDs)=0
Add an array of plug-in IDs to a property map.
- virtual [AAX_CBoolean](#) [GetPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPlugInIdentifierTriad](#) **outPluginIDs, uint32_t *outNumPluginIDs) const =0
Get an array of plug-in IDs from a property map.

Public Member Functions inherited from [AAX_IACFPropertyMap](#)

- virtual [AAX_CBoolean](#) [GetProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) *outValue) const =0
Get a property value from a property map.
- virtual [AAX_Result](#) [AddProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) inValue)=0
Add a property to a property map.
- virtual [AAX_Result](#) [RemoveProperty](#) ([AAX_EProperty](#) inProperty)=0
Remove a property from a property map.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.83.2 Member Function Documentation

14.83.2.1 GetProperty64()

```
virtual AAX\_CBoolean AAX_IACFPropertyMap_V3::GetProperty64 (
    AAX\_EProperty inProperty,
    AAX\_CPropertyValue64 * outValue ) const [pure virtual]
```

Get a property value from a property map.

Returns true if the selected property is supported, false if it is not

Parameters

in	<i>inProperty</i>	The property ID
out	<i>outValue</i>	The property value

14.83.2.2 AddProperty64()

```
virtual AAX\_Result AAX_IACFPropertyMap_V3::AddProperty64 (
    AAX\_EProperty inProperty,
    AAX\_CPropertyValue64 inValue ) [pure virtual]
```

Add a property to a property map.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

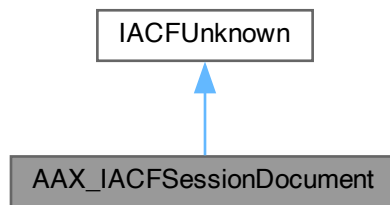
The documentation for this class was generated from the following file:

- [AAX_IACFPropertyMap.h](#)

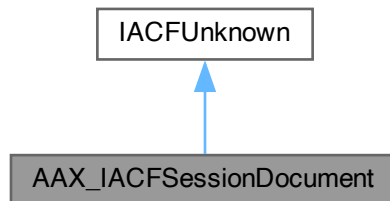
14.84 AAX_IACFSessionDocument Class Reference

```
#include <AAX_IACFSessionDocument.h>
```

Inheritance diagram for AAX_IACFSessionDocument:



Collaboration diagram for AAX_IACFSessionDocument:



14.84.1 Description

Interface representing information in a host session document.

Plug-in implementations should use [AAX_ISessionDocument](#) , which provides specific convenience methods for supported data types.

Public Member Functions

- virtual [AAX_Result](#) `GetDocumentData` ([AAX_DocumentData_UID](#) const &inDataType, [IACFUnknown](#) **outData)=0
Get data from the document.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.84.2 Member Function Documentation

14.84.2.1 GetDocumentData()

```
virtual AAX\_Result AAX_IACFSessionDocument::GetDocumentData (
    AAX\_DocumentData\_UID const & inDataType,
    IACFUnknown ** outData ) [pure virtual]
```

Get data from the document.

Get document data of a generic type

Similar to [QueryInterface\(\)](#) but uses a data type identifier rather than a true IID

The provided interface has already had a reference added, so be careful not to add an additional reference:

```
ACFPtr<MyType> ptr;
IACFUnknown * docDataPtr(nullptr);
if (AAX_SUCCESS == doc->GetDocumentData(dataUID, &docDataPtr) && docDataPtr) {
    ptr.attach(std::static_cast<MyType*>(docDataPtr)); // attach does not AddRef
}
```

Parameters

in	<i>inDataType</i>	The type of the document data requested
out	<i>outData</i>	An interface providing the requested data, or <code>nullptr</code> if the host does not support or cannot provide the requested data type. The reference count has been incremented on this object on behalf of the caller, so the caller must not add an additional reference count and must decrement the reference count on this object to release it. For information about which interface to expect for each requested data type, see the documentation for that data type.

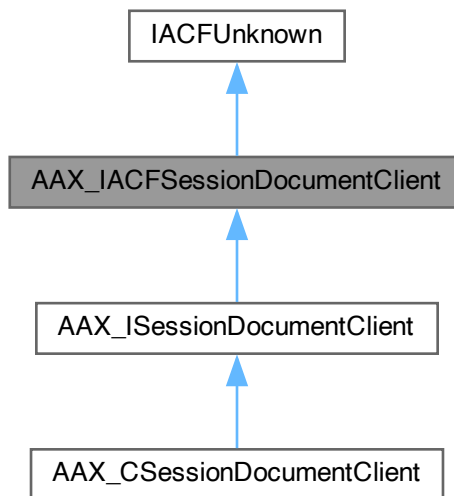
The documentation for this class was generated from the following file:

- [AAX_IACFSessionDocument.h](#)

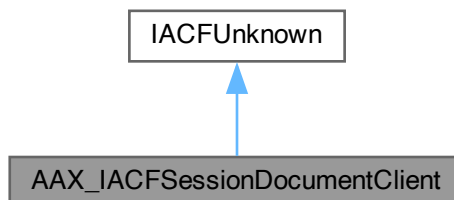
14.85 AAX_IACFSessionDocumentClient Class Reference

```
#include <AAX_IACFSessionDocumentClient.h>
```


Inheritance diagram for AAX_IACFSessionDocumentClient:



Collaboration diagram for AAX_IACFSessionDocumentClient:



14.85.1 Description

Interface representing a client of the session document interface.

For example, a plug-in implementation that makes calls on the session document interface provided by the host.

Public Member Functions

Initialization and uninitialization

- virtual `AAX_Result Initialize (IACFUnknown *iUnknown)=0`

- virtual [AAX_Result Uninitialize](#) (void)=0

Session document access

- virtual [AAX_Result SetSessionDocument](#) ([IACFUnknown](#) *iSessionDocument)=0
Sets or removes a session document.

AAX host and plug-in event notification

- virtual [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.85.2 Member Function Documentation

14.85.2.1 Initialize()

```
virtual AAX\_Result AAX_IACFSessionDocumentClient::Initialize (
    IACFUnknown * iUnknown ) [pure virtual]
```

Implemented in [AAX_CSessionDocumentClient](#).

14.85.2.2 Uninitialize()

```
virtual AAX\_Result AAX_IACFSessionDocumentClient::Uninitialize (
    void ) [pure virtual]
```

Implemented in [AAX_CSessionDocumentClient](#).

14.85.2.3 SetSessionDocument()

```
virtual AAX\_Result AAX_IACFSessionDocumentClient::SetSessionDocument (
    IACFUnknown * iSessionDocument ) [pure virtual]
```

Sets or removes a session document.

Parameters

in	<i>iSessionDocument</i>	Interface supporting at least AAX_IACFSessionDocument , or <code>nullptr</code> to indicate that any session document that is currently held should be released.
----	-------------------------	--

Implemented in [AAX_CSessionDocumentClient](#).

14.85.2.4 NotificationReceived()

```
virtual AAX\_Result AAX_IACFSessionDocumentClient::NotificationReceived (
    AAX\_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [pure virtual]
```

Notification Hook.

Called from the host to deliver notifications to this object.

Look at the [AAX_ENotificationEvent](#) enumeration to see a description of events you can listen for and the data they come with.

Note

- Different notifications are sent to different objects within a plug-in. If you are not seeing an expected notification, try checking the other plug-in objects' [NotificationReceived\(\)](#) methods.

Note

- the host may dispatch notifications synchronously or asynchronously, and calls to this method may occur concurrently on multiple threads.

A plug-in may also dispatch custom notifications using [AAX_IController::SendNotification\(\)](#). Custom notifications will be posted back to the plug-in's other objects which support a [NotificationReceived\(\)](#) method (e.g. the data model).

Parameters

in	<i>inNotificationType</i>	Type of notification being received. Notifications from the host are one of AAX_ENotificationEvent
in	<i>inNotificationData</i>	Block of incoming notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implemented in [AAX_CSessionDocumentClient](#).

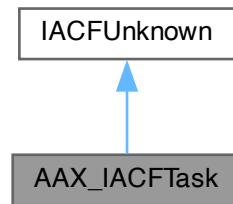
The documentation for this class was generated from the following file:

- [AAX_IACFSessionDocumentClient.h](#)

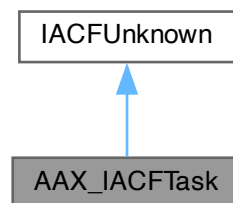
14.86 AAX_IACFTask Class Reference

```
#include <AAX_IACFTask.h>
```

Inheritance diagram for AAX_IACFTask:



Collaboration diagram for AAX_IACFTask:



14.86.1 Description

Versioned interface for an asynchronous task.

:Implemented by the AAX Host

Used by the [task agent](#).

This interface describes a task request and provides a way for the agent to express one or more results of the task as well as the progress of the task.

This interface is open-ended for both inputs and outputs. The host and agent must use common definitions for specific task types, their possible arguments, and the expected results.

Public Member Functions

- virtual [AAX_Result GetType](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_IACFDataBuffer](#) const * [GetArgumentOfType](#) ([AAX_CTypeID](#) iType) const =0
- virtual [AAX_Result SetProgress](#) (float iProgress)=0
- virtual float [GetProgress](#) () const =0
- virtual [AAX_Result AddResult](#) ([AAX_IACFDataBuffer](#) const *iResult)=0
 - Attach result data to this task.*
- virtual [AAX_Result SetDone](#) ([AAX_TaskCompletionStatus](#) iStatus)=0
 - Inform the host that the task is completed.*

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.86.2 Member Function Documentation**14.86.2.1 GetType()**

```
virtual AAX\_Result AAX_IACFTask::GetType (
    AAX\_CTypeID * oType ) const [pure virtual]
```

An identifier defining the type of the requested task

Parameters

out	<i>oType</i>	The type of this task request
-----	--------------	-------------------------------

14.86.2.2 GetArgumentOfType()

```
virtual AAX\_IACFDataBuffer const * AAX_IACFTask::GetArgumentOfType (
    AAX\_CTypeID iType ) const [pure virtual]
```

Additional information defining the request, depending on the task type

Parameters

in	<i>iType</i>	The type of argument requested. Possible argument types, if any, and the resulting data buffer format must be defined per task type.
----	--------------	--

Returns

The requested argument data, or nullptr. This data buffer's type ID is expected to match *iType* . The caller takes ownership of this object.

14.86.2.3 SetProgress()

```
virtual AAX_Result AAX_IACFTask::SetProgress (
    float iProgress ) [pure virtual]
```

Inform the host about the current status of the task

Parameters

in	<i>iProgress</i>	A value between 0 (no progress) and 1 (complete)
----	------------------	--

14.86.2.4 GetProgress()

```
virtual float AAX_IACFTask::GetProgress ( ) const [pure virtual]
```

Returns the current progress

14.86.2.5 AddResult()

```
virtual AAX_Result AAX_IACFTask::AddResult (
    AAX_IACFDataBuffer const * iResult ) [pure virtual]
```

Attach result data to this task.

This can be called multiple times to add multiple types of results to a single task.

The host may process the result data immediately or may wait for the task to complete.

The plug-in is expected to release the data buffer upon making this call. At a minimum, the data buffer must not be changed after this call is made. See `ACFPtr::inArg()`

Parameters

in	<i>iResult</i>	A buffer containing the result data. Expected result types, if any, and their data buffer format must be defined per task type.
----	----------------	---

14.86.2.6 SetDone()

```
virtual AAX_Result AAX_IACFTask::SetDone (
    AAX_TaskCompletionStatus iStatus ) [pure virtual]
```

Inform the host that the task is completed.

If `AAX_SUCCESS` is returned, the object should be considered invalid and released by the caller.

Parameters

in	<i>iStatus</i>	The final status of the task. This indicates to the host whether or not the task was performed as requested.
----	----------------	--

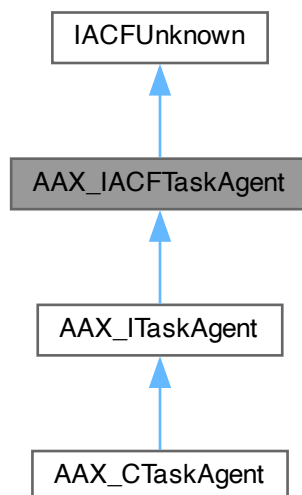
The documentation for this class was generated from the following file:

- [AAX_IACFTask.h](#)

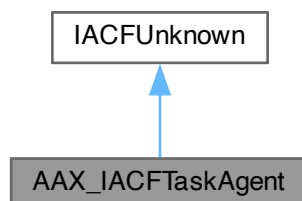
14.87 AAX_IACFTaskAgent Class Reference

```
#include <AAX_IACFTaskAgent.h>
```

Inheritance diagram for AAX_IACFTaskAgent:



Collaboration diagram for AAX_IACFTaskAgent:



14.87.1 Description

Versioned interface for a component that accepts task requests.

:Implemented by the Plug-In

The task agent is expected to complete the requested tasks asynchronously and to provide progress and completion details via calls on the [AAX_IACFTask](#) interface as the tasks proceed.

See also

[AAX_ITask](#)

Public Member Functions

Initialization and uninitialization

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
- virtual [AAX_Result Uninitialize](#) ()=0

Task management

- virtual [AAX_Result AddTask](#) ([IACFUnknown](#) *iTask)=0
- virtual [AAX_Result CancelAllTasks](#) ()=0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.87.2 Member Function Documentation

14.87.2.1 Initialize()

```
virtual AAX\_Result AAX_IACFTaskAgent::Initialize (
    IACFUnknown * iController ) [pure virtual]
```

Initialize the object

Parameters

in	<i>iController</i>	Interface allowing access to other objects in the object graph such as the plug-in's data model.
----	--------------------	--

Implemented in [AAX_CTaskAgent](#).

14.87.2.2 Uninitialize()

```
virtual AAX_Result AAX_IACFTaskAgent::Uninitialize ( ) [pure virtual]
```

Uninitialize the object

This method should release references to any shared objects

Implemented in [AAX_CTaskAgent](#).

14.87.2.3 AddTask()

```
virtual AAX_Result AAX_IACFTaskAgent::AddTask (
    IACFUnknown * iTask ) [pure virtual]
```

Request that the agent perform a task

Parameters

in	<i>iTask</i>	The task to perform. The agent must retain a reference to this task if it will be used beyond the scope of this method. This object should support at least AAX_IACFTask .
----	--------------	--

Implemented in [AAX_CTaskAgent](#).

14.87.2.4 CancelAllTasks()

```
virtual AAX_Result AAX_IACFTaskAgent::CancelAllTasks ( ) [pure virtual]
```

Request that the agent cancel all outstanding tasks

Implemented in [AAX_CTaskAgent](#).

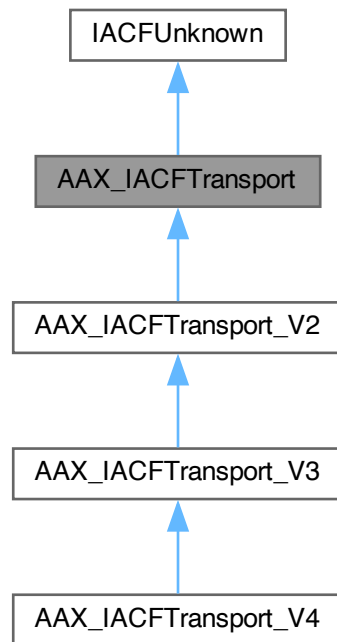
The documentation for this class was generated from the following file:

- [AAX_IACFTaskAgent.h](#)

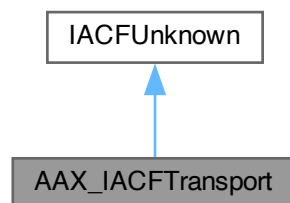
14.88 AAX_IACFTransport Class Reference

```
#include <AAX_IACFTransport.h>
```

Inheritance diagram for AAX_IACFTransport:



Collaboration diagram for AAX_IACFTransport:



14.88.1 Description

Versioned interface to get information about the host's transport state.

Public Member Functions

- virtual [AAX_Result GetCurrentTempo](#) (double *TempoBPM) const =0
CALL: Gets the current tempo.
- virtual [AAX_Result GetCurrentMeter](#) (int32_t *MeterNumerator, int32_t *MeterDenominator) const =0
CALL: Gets the current meter.
- virtual [AAX_Result IsTransportPlaying](#) (bool *isPlaying) const =0
CALL: Indicates whether or not the transport is playing back.
- virtual [AAX_Result GetCurrentTickPosition](#) (int64_t *TickPosition) const =0
CALL: Gets the current tick position.
- virtual [AAX_Result GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0
CALL: Gets current information on loop playback.
- virtual [AAX_Result GetCurrentNativeSampleLocation](#) (int64_t *SampleLocation) const =0
CALL: Gets the current playback location of the native audio engine.
- virtual [AAX_Result GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding tick position.
- virtual [AAX_Result GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t SampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding bar and beat position.
- virtual [AAX_Result GetTicksPerQuarter](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per quarter note.
- virtual [AAX_Result GetCurrentTicksPerBeat](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per beat.

Public Member Functions inherited from IACFUnknown

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const acfIID &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.88.2 Member Function Documentation

14.88.2.1 GetCurrentTempo()

```
virtual AAX\_Result AAX_IACFTransport::GetCurrentTempo (
    double * TempoBPM ) const [pure virtual]
```

CALL: Gets the current tempo.

Returns the tempo corresponding to the current position of the transport counter

Note

The resolution of the tempo returned here is based on the host's tempo resolution, so it will match the tempo displayed in the host. Use [GetCurrentTicksPerBeat\(\)](#) to calculate the tempo resolution note.

Parameters

out	<i>TempoBPM</i>	The current tempo in beats per minute
-----	-----------------	---------------------------------------

14.88.2.2 GetCurrentMeter()

```
virtual AAX_Result AAX_IACFTransport::GetCurrentMeter (
    int32_t * MeterNumerator,
    int32_t * MeterDenominator ) const [pure virtual]
```

CALL: Gets the current meter.

Returns the meter corresponding to the current position of the transport counter

Parameters

out	<i>MeterNumerator</i>	The numerator portion of the meter
out	<i>MeterDenominator</i>	The denominator portion of the meter

14.88.2.3 IsTransportPlaying()

```
virtual AAX_Result AAX_IACFTransport::IsTransportPlaying (
    bool * isPlaying ) const [pure virtual]
```

CALL: Indicates whether or not the transport is playing back.

Parameters

out	<i>isPlaying</i>	true if the transport is currently in playback
-----	------------------	--

14.88.2.4 GetCurrentTickPosition()

```
virtual AAX_Result AAX_IACFTransport::GetCurrentTickPosition (
    int64_t * TickPosition ) const [pure virtual]
```

CALL: Gets the current tick position.

Returns the current tick position corresponding to the current transport position. One "Tick" is represented here as 1/960000 of a quarter note. That is, there are 960,000 of these ticks in a quarter note.

Host Compatibility Notes The tick resolution here is different than that of the tick displayed in Pro Tools. "Display ticks" (as they are called) are 1/960 of a quarter note.

Parameters

out	<i>TickPosition</i>	The tick position value
-----	---------------------	-------------------------

14.88.2.5 GetCurrentLoopPosition()

```
virtual AAX_Result AAX_IACFTransport::GetCurrentLoopPosition (
    bool * bLooping,
    int64_t * LoopStartTick,
    int64_t * LoopEndTick ) const [pure virtual]
```

CALL: Gets current information on loop playback.

Host Compatibility Notes This does not indicate anything about the status of the "Loop Record" option. Even when the host is configured to loop playback, looping may not occur if certain conditions are not met (i.e. the length of the selection is too short)

Parameters

out	<i>bLooping</i>	true if the host is configured to loop playback
out	<i>LoopStartTick</i>	The starting tick position of the selection being looped (see GetCurrentTickPosition())
out	<i>LoopEndTick</i>	The ending tick position of the selection being looped (see GetCurrentTickPosition())

14.88.2.6 GetCurrentNativeSampleLocation()

```
virtual AAX_Result AAX_IACFTransport::GetCurrentNativeSampleLocation (
    int64_t * SampleLocation ) const [pure virtual]
```

CALL: Gets the current playback location of the native audio engine.

When called from a ProcessProc render callback, this method will provide the absolute sample location at the beginning of the callback's audio buffers.

When called from [AAX_IEffectParameters::RenderAudio_Hybrid\(\)](#), this method will provide the absolute sample location for the samples in the method's **output** audio buffers. To calculate the absolute sample location for the samples in the method's input buffers (i.e. the timeline location where the samples originated) subtract the value provided by [AAX_IController::GetHybridSignalLatency\(\)](#) from this value.

When called from a non-real-time thread, this method will provide the current location of the samples being processed by the plug-in's ProcessProc on its real-time processing thread.

Note

This method only returns a value during playback. It cannot be used to determine, e.g., the location of the timeline selector while the host is not in playback.

Parameters

out	<i>SampleLocation</i>	Absolute sample location of the first sample in the current native processing buffer
-----	-----------------------	--

14.88.2.7 GetCustomTickPosition()

```
virtual AAX_Result AAX_IACFTransport::GetCustomTickPosition (
    int64_t * oTickPosition,
    int64_t iSampleLocation ) const [pure virtual]
```

CALL: Given an absolute sample position, gets the corresponding tick position.

Host Compatibility Notes There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Parameters

out	<i>oTickPosition</i>	the timeline tick position corresponding to <i>iSampleLocation</i>
in	<i>iSampleLocation</i>	An absolute sample location (see GetCurrentNativeSampleLocation())

14.88.2.8 GetBarBeatPosition()

```
virtual AAX_Result AAX_IACFTransport::GetBarBeatPosition (
    int32_t * Bars,
    int32_t * Beats,
    int64_t * DisplayTicks,
    int64_t SampleLocation ) const [pure virtual]
```

CALL: Given an absolute sample position, gets the corresponding bar and beat position.

Host Compatibility Notes There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Parameters

out	<i>Bars</i>	The bar corresponding to <i>SampleLocation</i>
out	<i>Beats</i>	The beat corresponding to <i>SampleLocation</i>
out	<i>DisplayTicks</i>	The ticks corresponding to <i>SampleLocation</i>
in	<i>SampleLocation</i>	An absolute sample location (see GetCurrentNativeSampleLocation())

14.88.2.9 GetTicksPerQuarter()

```
virtual AAX_Result AAX_IACFTransport::GetTicksPerQuarter (
    uint32_t * ticks ) const [pure virtual]
```

CALL: Retrieves the number of ticks per quarter note.

Parameters

out	<i>ticks</i>	
-----	--------------	--

14.88.2.10 GetCurrentTicksPerBeat()

```
virtual AAX_Result AAX_IACFTransport::GetCurrentTicksPerBeat (
    uint32_t * ticks ) const [pure virtual]
```

CALL: Retrieves the number of ticks per beat.

Parameters

out	<i>ticks</i>	
-----	--------------	--

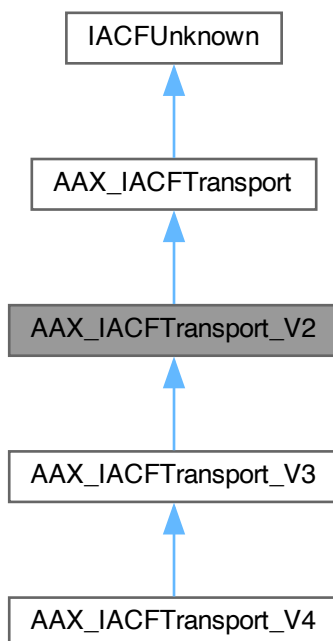
The documentation for this class was generated from the following file:

- [AAX_IACFTransport.h](#)

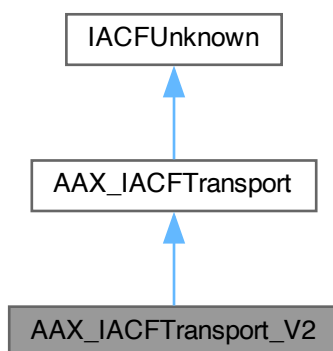
14.89 AAX_IACFTransport_V2 Class Reference

```
#include <AAX_IACFTransport.h>
```

Inheritance diagram for AAX_IACFTransport_V2:



Collaboration diagram for AAX_IACFTransport_V2:



14.89.1 Description

Versioned interface to get information about the host's transport state.

Public Member Functions

- virtual [AAX_Result GetTimelineSelectionStartPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the beginning of the current transport selection.
- virtual [AAX_Result GetTimeCodeInfo](#) ([AAX_EFrameRate](#) *oFrameRate, int32_t *oOffset) const =0
CALL: Retrieves the current time code frame rate and offset.
- virtual [AAX_Result GetFeetFramesInfo](#) ([AAX_EFeetFramesRate](#) *oFeetFramesRate, int64_t *oOffset) const =0
CALL: Retrieves the current timecode feet/frames rate and offset.
- virtual [AAX_Result IsMetronomeEnabled](#) (int32_t *isEnabled) const =0
Sets isEnabled to true if the metronome is enabled.

Public Member Functions inherited from [AAX_IACFTransport](#)

- virtual [AAX_Result GetCurrentTempo](#) (double *TempoBPM) const =0
CALL: Gets the current tempo.
- virtual [AAX_Result GetCurrentMeter](#) (int32_t *MeterNumerator, int32_t *MeterDenominator) const =0
CALL: Gets the current meter.
- virtual [AAX_Result IsTransportPlaying](#) (bool *isPlaying) const =0
CALL: Indicates whether or not the transport is playing back.
- virtual [AAX_Result GetCurrentTickPosition](#) (int64_t *TickPosition) const =0
CALL: Gets the current tick position.
- virtual [AAX_Result GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0
CALL: Gets current information on loop playback.
- virtual [AAX_Result GetCurrentNativeSampleLocation](#) (int64_t *SampleLocation) const =0
CALL: Gets the current playback location of the native audio engine.
- virtual [AAX_Result GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding tick position.
- virtual [AAX_Result GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t SampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding bar and beat position.
- virtual [AAX_Result GetTicksPerQuarter](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per quarter note.
- virtual [AAX_Result GetCurrentTicksPerBeat](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per beat.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual [acfUInt32](#) ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.89.2 Member Function Documentation

14.89.2.1 GetTimelineSelectionStartPosition()

```
virtual AAX_Result AAX_IACFTransport_V2::GetTimelineSelectionStartPosition (
    int64_t * oSampleLocation ) const [pure virtual]
```

CALL: Retrieves the absolute sample position of the beginning of the current transport selection.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oSampleLocation</i>	
-----	------------------------	--

14.89.2.2 GetTimeCodeInfo()

```
virtual AAX_Result AAX_IACFTransport_V2::GetTimeCodeInfo (
    AAX_EFrameRate * oFrameRate,
    int32_t * oOffset ) const [pure virtual]
```

CALL: Retrieves the current time code frame rate and offset.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oFrameRate</i>	
out	<i>oOffset</i>	

14.89.2.3 GetFeetFramesInfo()

```
virtual AAX_Result AAX_IACFTransport_V2::GetFeetFramesInfo (
    AAX_EFeetFramesRate * oFeetFramesRate,
    int64_t * oOffset ) const [pure virtual]
```

CALL: Retrieves the current timecode feet/frames rate and offset.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oFeetFramesRate</i>	
out	<i>oOffset</i>	

14.89.2.4 IsMetronomeEnabled()

```
virtual AAX\_Result AAX_IACFTransport_V2::IsMetronomeEnabled (
    int32_t * isEnabled ) const    [pure virtual]
```

Sets isEnabled to true if the metronome is enabled.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>isEnabled</i>	
-----	------------------	--

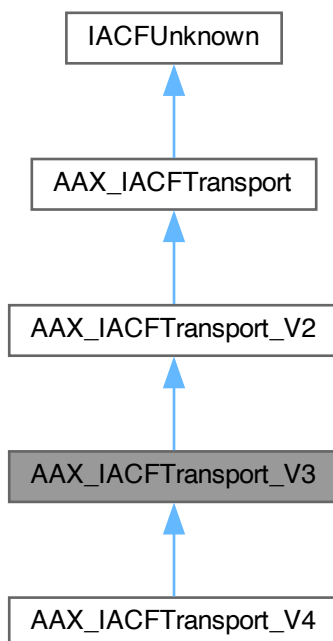
The documentation for this class was generated from the following file:

- [AAX_IACFTransport.h](#)

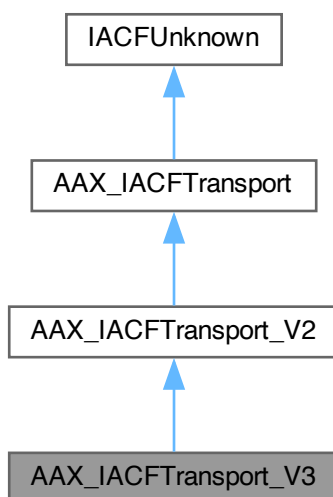
14.90 AAX_IACFTransport_V3 Class Reference

```
#include <AAX_IACFTransport.h>
```

Inheritance diagram for AAX_IACFTransport_V3:



Collaboration diagram for AAX_IACFTransport_V3:



14.90.1 Description

Versioned interface to get information about the host's transport state.

Public Member Functions

- virtual [AAX_Result GetHDTIMECodeInfo](#) ([AAX_EFrameRate](#) *oHDFrameRate, int64_t *oHDOffset) const =0
CALL: Retrieves the current HD time code frame rate and offset.

Public Member Functions inherited from [AAX_IACFTransport_V2](#)

- virtual [AAX_Result GetTimelineSelectionStartPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the beginning of the current transport selection.
- virtual [AAX_Result GetTIMECodeInfo](#) ([AAX_EFrameRate](#) *oFrameRate, int32_t *oOffset) const =0
CALL: Retrieves the current time code frame rate and offset.
- virtual [AAX_Result GetFeetFramesInfo](#) ([AAX_EFeetFramesRate](#) *oFeetFramesRate, int64_t *oOffset) const =0
CALL: Retrieves the current timecode feet/frames rate and offset.
- virtual [AAX_Result IsMetronomeEnabled](#) (int32_t *isEnabled) const =0
Sets isEnabled to true if the metronome is enabled.

Public Member Functions inherited from [AAX_IACFTransport](#)

- virtual [AAX_Result GetCurrentTempo](#) (double *TempoBPM) const =0
CALL: Gets the current tempo.
- virtual [AAX_Result GetCurrentMeter](#) (int32_t *MeterNumerator, int32_t *MeterDenominator) const =0
CALL: Gets the current meter.
- virtual [AAX_Result IsTransportPlaying](#) (bool *isPlaying) const =0
CALL: Indicates whether or not the transport is playing back.
- virtual [AAX_Result GetCurrentTickPosition](#) (int64_t *TickPosition) const =0
CALL: Gets the current tick position.
- virtual [AAX_Result GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0
CALL: Gets current information on loop playback.
- virtual [AAX_Result GetCurrentNativeSampleLocation](#) (int64_t *SampleLocation) const =0
CALL: Gets the current playback location of the native audio engine.
- virtual [AAX_Result GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding tick position.
- virtual [AAX_Result GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t SampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding bar and beat position.
- virtual [AAX_Result GetTicksPerQuarter](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per quarter note.
- virtual [AAX_Result GetCurrentTicksPerBeat](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per beat.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.90.2 Member Function Documentation

14.90.2.1 GetHDTimeCodeInfo()

```
virtual AAX\_Result AAX_IACFTransport_V3::GetHDTimeCodeInfo (
    AAX\_EFrameRate * oHDFrameRate,
    int64_t * oHDOffset ) const [pure virtual]
```

CALL: Retrieves the current HD time code frame rate and offset.

Note

This method is part of the [version 3 transport interface](#)

Parameters

out	<i>oHDFrameRate</i>	
out	<i>oHDOffset</i>	

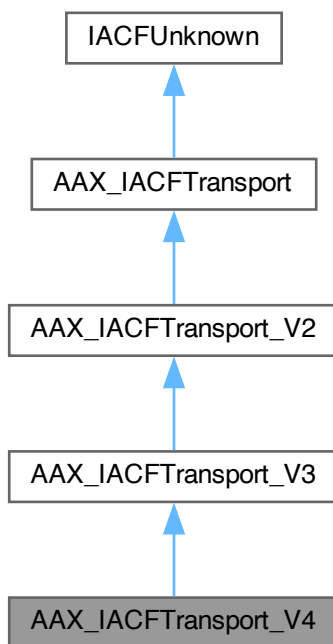
The documentation for this class was generated from the following file:

- [AAX_IACFTransport.h](#)

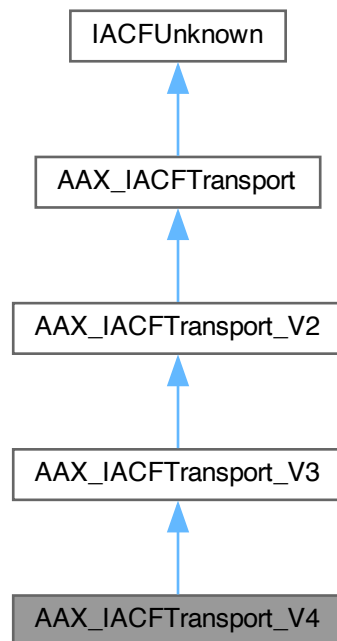
14.91 AAX_IACFTransport_V4 Class Reference

```
#include <AAX_IACFTransport.h>
```

Inheritance diagram for AAX_IACFTransport_V4:



Collaboration diagram for AAX_IACFTransport_V4:



14.91.1 Description

Versioned interface to get information about the host's transport state.

Public Member Functions

- virtual [AAX_Result GetTimelineSelectionEndPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the end of the current transport selection.

Public Member Functions inherited from [AAX_IACFTransport_V3](#)

- virtual [AAX_Result GetHDTIMECodeInfo](#) ([AAX_EFrameRate](#) *oHDFrameRate, int64_t *oHDOffset) const =0
CALL: Retrieves the current HD time code frame rate and offset.

Public Member Functions inherited from [AAX_IACFTransport_V2](#)

- virtual [AAX_Result GetTimelineSelectionStartPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the beginning of the current transport selection.
- virtual [AAX_Result GetTimeCodeInfo](#) ([AAX_EFrameRate](#) *oFrameRate, int32_t *oOffset) const =0
CALL: Retrieves the current time code frame rate and offset.
- virtual [AAX_Result GetFeetFramesInfo](#) ([AAX_EFeetFramesRate](#) *oFeetFramesRate, int64_t *oOffset) const =0
CALL: Retrieves the current timecode feet/frames rate and offset.
- virtual [AAX_Result IsMetronomeEnabled](#) (int32_t *isEnabled) const =0
Sets isEnabled to true if the metronome is enabled.

Public Member Functions inherited from [AAX_IACFTransport](#)

- virtual [AAX_Result GetCurrentTempo](#) (double *TempoBPM) const =0
CALL: Gets the current tempo.
- virtual [AAX_Result GetCurrentMeter](#) (int32_t *MeterNumerator, int32_t *MeterDenominator) const =0
CALL: Gets the current meter.
- virtual [AAX_Result IsTransportPlaying](#) (bool *isPlaying) const =0
CALL: Indicates whether or not the transport is playing back.
- virtual [AAX_Result GetCurrentTickPosition](#) (int64_t *TickPosition) const =0
CALL: Gets the current tick position.
- virtual [AAX_Result GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0
CALL: Gets current information on loop playback.
- virtual [AAX_Result GetCurrentNativeSampleLocation](#) (int64_t *SampleLocation) const =0
CALL: Gets the current playback location of the native audio engine.
- virtual [AAX_Result GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding tick position.
- virtual [AAX_Result GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t SampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding bar and beat position.
- virtual [AAX_Result GetTicksPerQuarter](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per quarter note.
- virtual [AAX_Result GetCurrentTicksPerBeat](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per beat.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.91.2 Member Function Documentation**14.91.2.1 [GetTimelineSelectionEndPosition\(\)](#)**

```
virtual AAX\_Result AAX_IACFTransport_V4::GetTimelineSelectionEndPosition (
    int64_t * oSampleLocation ) const [pure virtual]
```

CALL: Retrieves the absolute sample position of the end of the current transport selection.

Note

This method is part of the [version 4 transport interface](#)

Parameters

out	<i>oSampleLocation</i>	
-----	------------------------	--

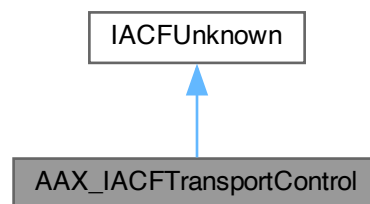
The documentation for this class was generated from the following file:

- [AAX_IACFTransport.h](#)

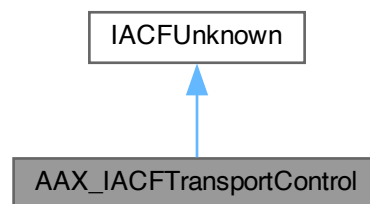
14.92 AAX_IACFTransportControl Class Reference

```
#include <AAX_IACFTransportControl.h>
```

Inheritance diagram for AAX_IACFTransportControl:



Collaboration diagram for AAX_IACFTransportControl:



14.92.1 Description

Versioned interface to control the host's transport state.

Public Member Functions

- virtual [AAX_Result RequestTransportStart](#) ()=0
CALL: Request that the host transport start playback.
- virtual [AAX_Result RequestTransportStop](#) ()=0
CALL: Request that the host transport stop playback.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.92.2 Member Function Documentation

14.92.2.1 RequestTransportStart()

```
virtual AAX\_Result AAX_IACFTransportControl::RequestTransportStart ( ) [pure virtual]
```

CALL: Request that the host transport start playback.

Note

This method is part of the [AAX_IACFTransportControl](#) interface

14.92.2.2 RequestTransportStop()

```
virtual AAX\_Result AAX_IACFTransportControl::RequestTransportStop ( ) [pure virtual]
```

CALL: Request that the host transport stop playback.

Note

This method is part of the [AAX_IACFTransportControl](#) interface

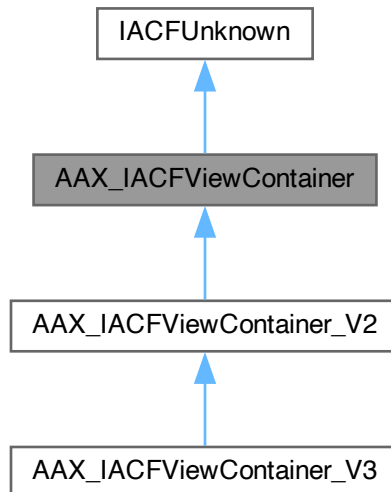
The documentation for this class was generated from the following file:

- [AAX_IACFTransportControl.h](#)

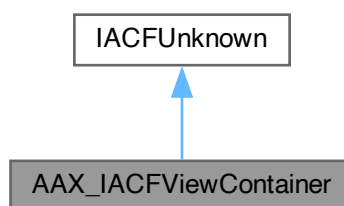
14.93 AAX_IACFViewContainer Class Reference

```
#include <AAX_IACFViewContainer.h>
```

Inheritance diagram for AAX_IACFViewContainer:



Collaboration diagram for AAX_IACFViewContainer:



14.93.1 Description

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the host app and by effect components.

See also

[AAX_IViewContainer](#)

Public Member Functions

View and GUI state queries

- virtual `int32_t GetType ()=0`
Returns the raw view type as one of [AAX_EViewContainer_Type](#).
- virtual `void * GetPtr ()=0`
Returns a pointer to the raw view.
- virtual `AAX_Result GetModifiers (uint32_t *outModifiers)=0`
Queries the host for the current [modifier keys](#).

View change requests

- virtual `AAX_Result SetViewSize (AAX_Point &inSize)=0`
Request a change to the main view size.

Host event handlers

- virtual `AAX_Result HandleParameterMouseDown (AAX_CParamID inParamID, uint32_t inModifiers)=0`
Alert the host to a mouse down event.
- virtual `AAX_Result HandleParameterMouseDrag (AAX_CParamID inParamID, uint32_t inModifiers)=0`
Alert the host to a mouse drag event.
- virtual `AAX_Result HandleParameterMouseUp (AAX_CParamID inParamID, uint32_t inModifiers)=0`
Alert the host to a mouse up event.

Public Member Functions inherited from [IACFUnknown](#)

- virtual `BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE QueryInterface (const acfIID &iid, void **ppOut)=0`
Returns pointers to supported interfaces.
- virtual `acfUInt32 ACFMETHODCALLTYPE AddRef (void)=0`
Increments reference count.
- virtual `acfUInt32 ACFMETHODCALLTYPE Release (void)=0`
Decrements reference count.

14.93.2 Member Function Documentation

14.93.2.1 GetType()

```
virtual int32_t AAX_IACFViewContainer::GetType ( ) [pure virtual]
```

Returns the raw view type as one of [AAX_EViewContainer_Type](#).

14.93.2.2 GetPtr()

```
virtual void * AAX_IACFViewContainer::GetPtr ( ) [pure virtual]
```

Returns a pointer to the raw view.

14.93.2.3 GetModifiers()

```
virtual AAX_Result AAX_IACFViewContainer::GetModifiers (
    uint32_t * outModifiers ) [pure virtual]
```

Queries the host for the current [modifier keys](#).

This method returns a bit mask with bits set for each of the currently active modifier keys. This method does not return the state of the [AAX_eModifiers_SecondaryButton](#).

Host Compatibility Notes Although this method allows plug-ins to acquire the current state of the Windows key (normally blocked by Pro Tools), plug-ins should not use key combinations that require this key.

Parameters

out	<i>outModifiers</i>	Current modifiers as a bitmask of AAX_EModifiers
-----	---------------------	--

14.93.2.4 SetViewSize()

```
virtual AAX_Result AAX_IACFViewContainer::SetViewSize (
    AAX_Point & inSize ) [pure virtual]
```

Request a change to the main view size.

Note

- For compatibility with the smallest supported displays, plug-in GUI dimensions should not exceed 749x617 pixels, or 749x565 pixels for plug-ins with sidechain support.

Parameters

in	<i>inSize</i>	The new size to which the plug-in view should be set
----	---------------	--

14.93.2.5 HandleParameterMouseDown()

```
virtual AAX_Result AAX_IACFViewContainer::HandleParameterMouseDown (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse down event.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

14.93.2.6 HandleParameterMouseDrag()

```
virtual AAX_Result AAX_IACFViewContainer::HandleParameterMouseDrag (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse drag event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209 / PT-218474](#))

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

14.93.2.7 HandleParameterMouseUp()

```
virtual AAX_Result AAX_IACFViewContainer::HandleParameterMouseUp (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse up event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209 / PT-218474](#))

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

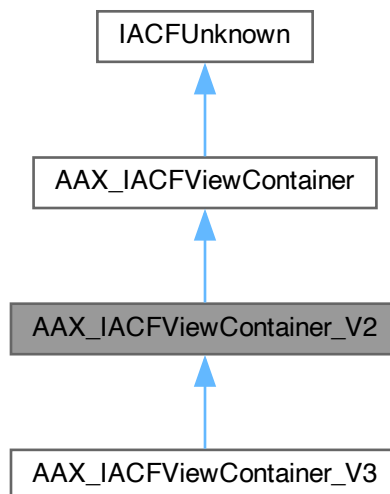
The documentation for this class was generated from the following file:

- [AAX_IACFViewContainer.h](#)

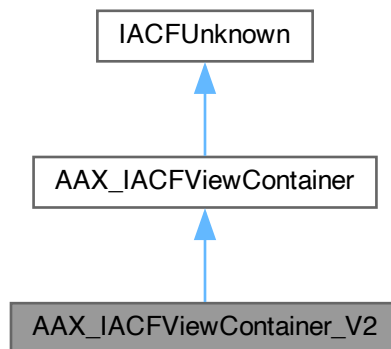
14.94 AAX_IACFViewContainer_V2 Class Reference

```
#include <AAX_IACFViewContainer.h>
```

Inheritance diagram for AAX_IACFViewContainer_V2:



Collaboration diagram for AAX_IACFViewContainer_V2:



14.94.1 Description

Supplemental interface for the AAX host's view of a single instance of an effect. Used by both clients of the host app and by effect components.

See also

[AAX_IViewContainer](#)

Public Member Functions

Host event handlers

- virtual [AAX_Result HandleMultipleParametersMouseDown](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse down event.
- virtual [AAX_Result HandleMultipleParametersMouseDrag](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse drag event.
- virtual [AAX_Result HandleMultipleParametersMouseUp](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse up event.

Public Member Functions inherited from [AAX_IACFViewContainer](#)

- virtual int32_t [GetType](#) ()=0
Returns the raw view type as one of [AAX_EViewContainer_Type](#).
- virtual void * [GetPtr](#) ()=0
Returns a pointer to the raw view.
- virtual [AAX_Result GetModifiers](#) (uint32_t *outModifiers)=0
Queries the host for the current [modifier keys](#).

- virtual [AAX_Result SetViewSize](#) ([AAX_Point](#) &inSize)=0
Request a change to the main view size.
- virtual [AAX_Result HandleParameterMouseDown](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse down event.
- virtual [AAX_Result HandleParameterMouseDrag](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse drag event.
- virtual [AAX_Result HandleParameterMouseUp](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse up event.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.94.2 Member Function Documentation

14.94.2.1 HandleMultipleParametersMouseDown()

```
virtual AAX\_Result AAX_IACFViewContainer_V2::HandleMultipleParametersMouseDown (
    const AAX\_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse down event.

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

14.94.2.2 HandleMultipleParametersMouseDrag()

```
virtual AAX\_Result AAX_IACFViewContainer_V2::HandleMultipleParametersMouseDrag (
    const AAX\_CParamID * inParamIDs,
```

```
uint32_t inNumOfParams,
uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse drag event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

14.94.2.3 HandleMultipleParametersMouseUp()

```
virtual AAX\_Result AAX_IACFViewContainer_V2::HandleMultipleParametersMouseUp (
    const AAX\_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse up event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

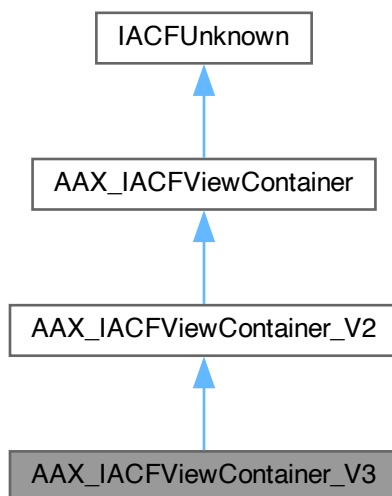
The documentation for this class was generated from the following file:

- [AAX_IACFViewContainer.h](#)

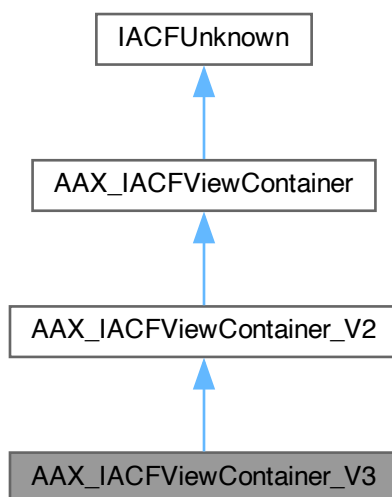
14.95 AAX_IACFViewContainer_V3 Class Reference

```
#include <AAX_IACFViewContainer.h>
```

Inheritance diagram for AAX_IACFViewContainer_V3:



Collaboration diagram for AAX_IACFViewContainer_V3:



14.95.1 Description

Additional methods to track mouse as it moves over controls.

See also

[AAX_IViewContainer](#)

Public Member Functions

Host event handlers

- virtual [AAX_Result HandleParameterMouseEnter](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse enter event to the parameter's control.
- virtual [AAX_Result HandleParameterMouseExit](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse exit event from the parameter's control.

Public Member Functions inherited from [AAX_IACFViewContainer_V2](#)

- virtual [AAX_Result HandleMultipleParametersMouseDown](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse down event.
- virtual [AAX_Result HandleMultipleParametersMouseDrag](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse drag event.
- virtual [AAX_Result HandleMultipleParametersMouseUp](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse up event.

Public Member Functions inherited from [AAX_IACFViewContainer](#)

- virtual int32_t [GetType](#) ()=0
Returns the raw view type as one of [AAX_EViewContainer_Type](#).
- virtual void * [GetPtr](#) ()=0
Returns a pointer to the raw view.
- virtual [AAX_Result GetModifiers](#) (uint32_t *outModifiers)=0
Queries the host for the current [modifier keys](#).
- virtual [AAX_Result SetViewSize](#) ([AAX_Point](#) &inSize)=0
Request a change to the main view size.
- virtual [AAX_Result HandleParameterMouseDown](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse down event.
- virtual [AAX_Result HandleParameterMouseDrag](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse drag event.
- virtual [AAX_Result HandleParameterMouseUp](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse up event.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.95.2 Member Function Documentation

14.95.2.1 HandleParameterMouseEnter()

```
virtual AAX_Result AAX_IACFViewContainer_V3::HandleParameterMouseEnter (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse enter event to the parameter's control.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being entered
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Returns AAX_SUCCESS if event was processed successfully, otherwise an AAX_ERROR code

14.95.2.2 HandleParameterMouseExit()

```
virtual AAX_Result AAX_IACFViewContainer_V3::HandleParameterMouseExit (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse exit event from the parameter's control.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being exited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Returns AAX_SUCCESS if event was processed successfully, otherwise an AAX_ERROR code

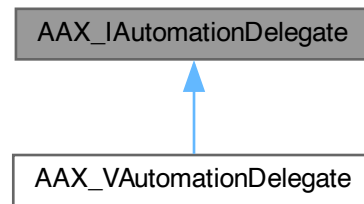
The documentation for this class was generated from the following file:

- [AAX_IACFViewContainer.h](#)

14.96 AAX_IAutomationDelegate Class Reference

```
#include <AAX_IAutomationDelegate.h>
```

Inheritance diagram for AAX_IAutomationDelegate:



14.96.1 Description

Interface allowing an AAX plug-in to interact with the host's event system.

:Implemented by the AAX Host

This delegate provides a means of interacting with the host's event system in order to ensure that events such as parameter updates are properly arbitrated and broadcast to all listeners. The automation delegate is used regardless of whether or not an individual parameter is "automatable" or "automation-enabled".

A parameter must be registered with the automation delegate in order for updates to the parameter's control in the plug-in's GUI or other controller (control surface, etc.) to be successfully processed by the host and sent to the [AAX_IEffectParameters](#) object.

The parameter identifiers used by this interface correspond to the control IDs used to identify parameters in the [Parameter Manager](#).

Public Member Functions

- virtual [~AAX_IAutomationDelegate](#) ()
- virtual [AAX_Result RegisterParameter](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result UnregisterParameter](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result PostSetValueRequest](#) ([AAX_CParamID](#) iParameterID, double normalizedValue) const =0
- virtual [AAX_Result PostCurrentValue](#) ([AAX_CParamID](#) iParameterID, double normalizedValue) const =0
- virtual [AAX_Result PostTouchRequest](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result PostReleaseRequest](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result GetTouchState](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *oTouched)=0
- virtual [AAX_Result ParameterNameChanged](#) ([AAX_CParamID](#) iParameterID)=0

14.96.2 Constructor & Destructor Documentation

14.96.2.1 ~AAX_IAutomationDelegate()

```
virtual AAX_IAutomationDelegate::~~AAX_IAutomationDelegate ( ) [inline], [virtual]
```

14.96.3 Member Function Documentation

14.96.3.1 RegisterParameter()

```
virtual AAX_Result AAX_IAutomationDelegate::RegisterParameter (
    AAX_CParamID iParameterID ) [pure virtual]
```

Register a control with the automation system using a char* based control identifier

The automation delegate owns a list of the IDs of all of the parameters that have been registered with it. This list is used to set up listeners for all of the registered parameters such that the automation delegate may update the plug-in when the state of any of the registered parameters have been modified.

See also

[AAX_IAutomationDelegate::UnregisterParameter\(\)](#)

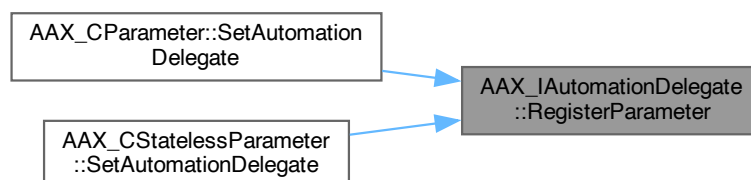
Parameters

in	<i>iParameterID</i>	Parameter ID that is being registered
----	---------------------	---------------------------------------

Implemented in [AAX_VAutomationDelegate](#).

Referenced by [AAX_CParameter< T >::SetAutomationDelegate\(\)](#), and [AAX_CStatelessParameter::SetAutomationDelegate\(\)](#).

Here is the caller graph for this function:



14.96.3.2 UnregisterParameter()

```
virtual AAX_Result AAX_IAutomationDelegate::UnregisterParameter (
    AAX_CParamID iParameterID ) [pure virtual]
```

Unregister a control with the automation system using a char* based control identifier

Note

All registered controls should be unregistered or the system might leak.

See also

[AAX_IAutomationDelegate::RegisterParameter\(\)](#)

Parameters

in	<i>iParameterID</i>	Parameter ID that is being registered
----	---------------------	---------------------------------------

Implemented in [AAX_VAutomationDelegate](#).

Referenced by [AAX_CStatelessParameter::SetAutomationDelegate\(\)](#).

Here is the caller graph for this function:



14.96.3.3 PostSetValueRequest()

```
virtual AAX_Result AAX_IAutomationDelegate::PostSetValueRequest (
    AAX_CParamID iParameterID,
    double normalizedValue ) const [pure virtual]
```

Submits a request for the given parameter's value to be changed

Parameters

in	<i>iParameterID</i>	ID of the parameter for which a change is requested
in	<i>normalizedValue</i>	The requested new parameter value, formatted as a double and normalized to [0 1]

Implemented in [AAX_VAutomationDelegate](#).

14.96.3.4 PostCurrentValue()

```
virtual AAX_Result AAX_IAutomationDelegate::PostCurrentValue (
    AAX_CParamID iParameterID,
    double normalizedValue ) const [pure virtual]
```

Notifies listeners that a parameter's value has changed

Parameters

in	<i>iParameterID</i>	ID of the parameter that has been updated
in	<i>normalizedValue</i>	The current parameter value, formatted as a double and normalized to [0 1]

Implemented in [AAX_VAutomationDelegate](#).

14.96.3.5 PostTouchRequest()

```
virtual AAX_Result AAX_IAutomationDelegate::PostTouchRequest (
    AAX_CParamID iParameterID ) [pure virtual]
```

Requests that the given parameter be "touched", i.e. locked for updates by the current client

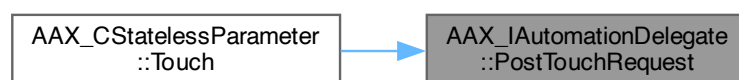
Parameters

in	<i>iParameterID</i>	ID of the parameter that will be touched
----	---------------------	--

Implemented in [AAX_VAutomationDelegate](#).

Referenced by [AAX_CStatelessParameter::Touch\(\)](#).

Here is the caller graph for this function:



14.96.3.6 PostReleaseRequest()

```
virtual AAX_Result AAX_IAutomationDelegate::PostReleaseRequest (
    AAX_CParamID iParameterID ) [pure virtual]
```

Requests that the given parameter be "released", i.e. available for updates from any client

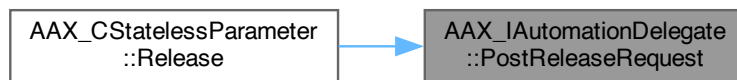
Parameters

in	<i>iParameterID</i>	ID of the parameter that will be released
----	---------------------	---

Implemented in [AAX_VAutomationDelegate](#).

Referenced by [AAX_CStatelessParameter::Release\(\)](#).

Here is the caller graph for this function:

**14.96.3.7 GetTouchState()**

```
virtual AAX_Result AAX_IAutomationDelegate::GetTouchState (
    AAX_CParamID iParameterID,
    AAX_CBoolean * oTouched ) [pure virtual]
```

Gets the current touched state of a parameter

Parameters

in	<i>iParameterID</i>	ID of the parameter that is being queried
out	<i>oTouched</i>	The current touch state of the parameter

Implemented in [AAX_VAutomationDelegate](#).

14.96.3.8 ParameterNameChanged()

```
virtual AAX_Result AAX_IAutomationDelegate::ParameterNameChanged (
    AAX_CParamID iParameterID ) [pure virtual]
```

Notify listeners that the parameter's display name has changed

Note that this is not part of the underlying automation delegate interface with the host; it is converted on the AAX side to a notification posted to the host via the [AAX_IController](#) .

Parameters

in	<i>iParameterID</i>	ID of the parameter that has been updated
----	---------------------	---

Implemented in [AAX_VAutomationDelegate](#).

Referenced by [AAX_CStatelessParameter::SetName\(\)](#).

Here is the caller graph for this function:



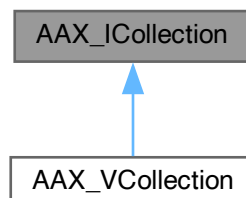
The documentation for this class was generated from the following file:

- [AAX_IAutomationDelegate.h](#)

14.97 AAX_ICollection Class Reference

```
#include <AAX_ICollection.h>
```

Inheritance diagram for AAX_ICollection:



14.97.1 Description

Interface to represent a plug-in binary's static description.

:Implemented by the AAX Host

The [AAX_ICollection](#) interface provides a creation function for new plug-in descriptors, which in turn provides access to the various interfaces necessary for describing a plug-in. When a plug-in description is complete, it is added to the collection via the [AddEffect](#) method. The [AAX_ICollection](#) interface also provides some additional description methods that are used to describe the overall plug-in package. These methods can be used to describe the plug-in package's name, the name of the plug-in's manufacturer, and the plug-in package version.

Legacy Porting Notes The information in [AAX_ICollection](#) is roughly analogous to the information provided by CProcessGroup in the legacy plug-in library

Public Member Functions

- virtual [~AAX_ICollection](#) ()
- virtual [AAX_IEffectDescriptor](#) * [NewDescriptor](#) ()=0
Create a new Effect descriptor.
- virtual [AAX_Result](#) [AddEffect](#) (const char *inEffectID, [AAX_IEffectDescriptor](#) *inEffectDescriptor)=0
Add an Effect description to the collection.
- virtual [AAX_Result](#) [SetManufacturerName](#) (const char *inPackageName)=0
Set the plug-in manufacturer name.
- virtual [AAX_Result](#) [AddPackageName](#) (const char *inPackageName)=0
Set the plug-in package name.
- virtual [AAX_Result](#) [SetPackageVersion](#) (uint32_t inVersion)=0
Set the plug-in package version number.
- virtual [AAX_IPropertyMap](#) * [NewPropertyMap](#) ()=0
Create a new property map.
- virtual [AAX_Result](#) [SetProperties](#) ([AAX_IPropertyMap](#) *inProperties)=0
Set the properties of the collection.
- virtual [AAX_IDescriptionHost](#) * [DescriptionHost](#) ()=0
- virtual const [AAX_IDescriptionHost](#) * [DescriptionHost](#) () const =0
- virtual [IACFDefinition](#) * [HostDefinition](#) () const =0

14.97.2 Constructor & Destructor Documentation

14.97.2.1 ~AAX_ICollection()

```
virtual AAX_ICollection::~~AAX_ICollection ( ) [inline], [virtual]
```

14.97.3 Member Function Documentation

14.97.3.1 NewDescriptor()

```
virtual AAX_IEffectDescriptor * AAX_ICollection::NewDescriptor ( ) [pure virtual]
```

Create a new Effect descriptor.

Implemented in [AAX_VCollection](#).

14.97.3.2 AddEffect()

```
virtual AAX_Result AAX_ICollection::AddEffect (
    const char * inEffectID,
    AAX_IEffectDescriptor * inEffectDescriptor ) [pure virtual]
```

Add an Effect description to the collection.

Each Effect that a plug-in registers with [AAX_ICollection::AddEffect\(\)](#) is considered a completely different user-facing product. For example, in Avid's Dynamics III plug-in the Expander, Compressor, and DeEsser are each registered as separate Effects. All stem format variations within each Effect are registered within that Effect's [AAX_IEffectDescriptor](#) using [AddComponent\(\)](#).

The [AAX_eProperty_ProductID](#) value for all ProcessProcs within a single Effect must be identical.

This method passes ownership of an [AAX_IEffectDescriptor](#) object to the [AAX_ICollection](#). The [AAX_IEffectDescriptor](#) must not be deleted by the AAX plug-in, nor should it be edited in any way after it is passed to the [AAX_ICollection](#).

Parameters

in	<i>inEffectID</i>	The effect ID.
in	<i>inEffectDescriptor</i>	The Effect descriptor.

Implemented in [AAX_VCollection](#).

14.97.3.3 SetManufacturerName()

```
virtual AAX_Result AAX_ICollection::SetManufacturerName (
    const char * inPackageName ) [pure virtual]
```

Set the plug-in manufacturer name.

Parameters

in	<i>inPackageName</i>	The name of the manufacturer.
----	----------------------	-------------------------------

Implemented in [AAX_VCollection](#).

14.97.3.4 AddPackageName()

```
virtual AAX_Result AAX_ICollection::AddPackageName (
    const char * inPackageName ) [pure virtual]
```

Set the plug-in package name.

May be called multiple times to add abbreviated package names.

Note

Every plug-in must include at least one name variant with 16 or fewer characters, plus a null terminating character. Used for Presets folder.

Parameters

in	<i>inPackageName</i>	The name of the package.
----	----------------------	--------------------------

Implemented in [AAX_VCollection](#).

14.97.3.5 SetPackageVersion()

```
virtual AAX_Result AAX_ICollection::SetPackageVersion (
    uint32_t inVersion ) [pure virtual]
```

Set the plug-in package version number.

Parameters

in	<i>inVersion</i>	The package version number.
----	------------------	-----------------------------

Implemented in [AAX_VCollection](#).

14.97.3.6 NewPropertyMap()

```
virtual AAX_IPropertyMap * AAX_ICollection::NewPropertyMap ( ) [pure virtual]
```

Create a new property map.

Implemented in [AAX_VCollection](#).

14.97.3.7 SetProperties()

```
virtual AAX_Result AAX_ICollection::SetProperties (
    AAX_IPropertyMap * inProperties ) [pure virtual]
```

Set the properties of the collection.

Parameters

in	<i>inProperties</i>	Collection properties
----	---------------------	-----------------------

Implemented in [AAX_VCollection](#).

14.97.3.8 DescriptionHost() [1/2]

```
virtual AAX\_IDescriptionHost * AAX_ICollection::DescriptionHost ( ) [pure virtual]
```

Get a pointer to an [AAX_IDescriptionHost](#), if supported by the host

This interface is served by the [AAX_ICollection](#) in order to avoid requiring a new method prototype for the [GetEffectDescriptions\(\)](#) method called from the AAX Library.

See also

[AAX_UIDs.h](#) for available feature UIDs, e.g. [AAXATTR_ClientFeature_AuxOutputStem](#)

Implemented in [AAX_VCollection](#).

14.97.3.9 DescriptionHost() [2/2]

```
virtual const AAX\_IDescriptionHost * AAX_ICollection::DescriptionHost ( ) const [pure virtual]
```

Get a pointer to an [AAX_IDescriptionHost](#), if supported by the host

This interface is served by the [AAX_ICollection](#) in order to avoid requiring a new method prototype for the [GetEffectDescriptions\(\)](#) method called from the AAX Library.

See also

[AAX_UIDs.h](#) for available feature UIDs, e.g. [AAXATTR_ClientFeature_AuxOutputStem](#)

Implemented in [AAX_VCollection](#).

14.97.3.10 HostDefinition()

```
virtual IACFDefinition * AAX_ICollection::HostDefinition ( ) const [pure virtual]
```

Get a pointer to an [IACFDefinition](#), if supported by the host

This interface is served by the [AAX_ICollection](#) in order to avoid requiring a new method prototype for the [GetEffectDescriptions\(\)](#) method called from the AAX Library.

See also

[AAX_UIDs.h](#) for available host attribute UUIDs, e.g. [AAXATTR_Client_Level](#)

The implementation of [AAX_ICollection](#) owns the referenced object. No AddRef occurs.

[IACFDefinition::DefineAttribute\(\)](#) is not supported on this object

Implemented in [AAX_VCollection](#).

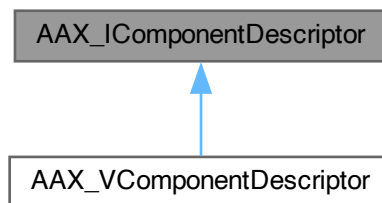
The documentation for this class was generated from the following file:

- [AAX_ICollection.h](#)

14.98 AAX_IComponentDescriptor Class Reference

```
#include <AAX_IComponentDescriptor.h>
```

Inheritance diagram for AAX_IComponentDescriptor:



14.98.1 Description

Description interface for an AAX plug-in component.

:Implemented by the AAX Host

This is an abstract interface containing everything needed to describe a single algorithm of an Effect. For more information about algorithm processing in AAX plug-ins, see [Real-time algorithm callback](#).

Public Member Functions

- virtual [~AAX_IComponentDescriptor](#) ()
- virtual [AAX_Result Clear](#) ()=0
Clears the descriptor.
- virtual [AAX_Result AddAudioIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio input context field.
- virtual [AAX_Result AddAudioOut](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio output context field.
- virtual [AAX_Result AddAudioBufferLength](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a buffer length context field.
- virtual [AAX_Result AddSampleRate](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a sample rate context field.
- virtual [AAX_Result AddClock](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a clock context field.
- virtual [AAX_Result AddSideChainIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a side-chain input context field.
- virtual [AAX_Result AddDataInPort](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inPacketSize, [AAX_EDataInPortType](#) inPortType=[AAX_eDataInPortType_Buffered](#))=0
Adds a custom data port to the algorithm context.
- virtual [AAX_Result AddAuxOutputStem](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inStemFormat, [const char](#) inNameUTF8[])=0
Adds an auxiliary output stem for a plug-in.
- virtual [AAX_Result AddPrivateData](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inDataSize, [uint32_t](#) inOptions=[AAX_ePrivateDataOptions_DefaultOptions](#))=0
Adds a private data port to the algorithm context.
- virtual [AAX_Result AddTemporaryData](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inDataElementSize)=0
Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.
- virtual [AAX_Result AddDmaInstance](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_IDma::EMode](#) inDmaMode)=0
Adds a DMA field to the plug-in's context.
- virtual [AAX_Result AddMeters](#) ([AAX_CFieldIndex](#) inFieldIndex, [const AAX_CTypeID](#) *inMeterIDs, [const uint32_t](#) inMeterCount)=0
Adds a meter field to the plug-in's context.
- virtual [AAX_Result AddMIDINode](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_EMIDINodeType](#) inNodeType, [const char](#) inNodeName[], [uint32_t](#) channelMask)=0
Adds a MIDI node field to the plug-in's context.
- virtual [AAX_Result AddReservedField](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inFieldType)=0
Subscribes a context field to host-provided services or information.
- virtual [AAX_IPropertyMap * NewPropertyMap](#) () [const](#) =0
Creates a new, empty property map.
- virtual [AAX_IPropertyMap * DuplicatePropertyMap](#) ([AAX_IPropertyMap](#) *inPropertyMap) [const](#) =0
Creates a new property map using an existing property map.
- virtual [AAX_Result AddProcessProc_Native](#) ([AAX_CProcessProc](#) inProcessProc, [AAX_IPropertyMap](#) *inProperties=NULL, [AAX_CInstanceInitProc](#) inInstanceInitProc=NULL, [AAX_CBackgroundProc](#) inBackgroundProc=NULL, [AAX_CSelector](#) *outProcID=NULL)=0
Registers an algorithm processing entrypoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc_TI](#) ([const char](#) inDLLFileNameUTF8[], [const char](#) inProcessProcSymbol[], [AAX_IPropertyMap](#) *inProperties, [const char](#) inInstanceInitProcSymbol[]=NULL, [const char](#) inBackgroundProcSymbol[]=NULL, [AAX_CSelector](#) *outProcID=NULL)=0
Registers an algorithm processing entrypoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc](#) ([AAX_IPropertyMap](#) *inProperties, [AAX_CSelector](#) *outProcIDs=NULL, [int32_t](#) inProcIDsSize=0)=0

Registers one or more algorithm processing entrypoints (process procedures)

- `template<typename aContextType >`
`AAX_Result AddProcessProc_Native` (void([AAX_CALLBACK](#) *inProcessProc)(aContextType *const inInstancesBegin[], const void *inInstancesEnd), [AAX_IPropertyMap](#) *inProperties=NULL, int32_t([AAX_CALLBACK](#) *inInstanceInitProc)(const aContextType *inInstanceContextPtr, [AAX_EComponentInstanceInitAction](#) inAction)=NULL, int32_t([AAX_CALLBACK](#) *inBackgroundProc)(void)=NULL)

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

14.98.2 Constructor & Destructor Documentation

14.98.2.1 ~AAX_IComponentDescriptor()

```
virtual AAX_IComponentDescriptor::~~AAX_IComponentDescriptor ( ) [inline], [virtual]
```

14.98.3 Member Function Documentation

14.98.3.1 Clear()

```
virtual AAX\_Result AAX_IComponentDescriptor::Clear ( ) [pure virtual]
```

Clears the descriptor.

Clears the descriptor and readies it for the next algorithm description

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.2 AddAudioIn()

```
virtual AAX\_Result AAX_IComponentDescriptor::AddAudioIn (
    AAX\_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes an audio input context field.

Defines an audio in port for host-provided information in the algorithm's context structure.

- Data type: float**
- Data kind: An array of float arrays, one for each input channel

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.3 AddAudioOut()

```
virtual AAX\_Result AAX_IComponentDescriptor::AddAudioOut (
    AAX\_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes an audio output context field.

Defines an audio out port for host-provided information in the algorithm's context structure.

- Data type: float**
- Data kind: An array of float arrays, one for each output channel

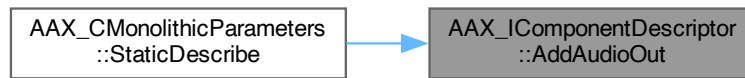
Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.4 AddAudioBufferLength()

```
virtual AAX_Result AAX_IComponentDescriptor::AddAudioBufferLength (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a buffer length context field.

Defines a buffer length port for host-provided information in the algorithm's context structure.

- Data type: `int32_t*`
- Data kind: The number of samples in the current audio buffer

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.5 AddSampleRate()

```
virtual AAX_Result AAX_IComponentDescriptor::AddSampleRate (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a sample rate context field.

Defines a sample rate port for host-provided information in the algorithm's context structure.

- Data type: [AAX_CSampleRate](#) *
- Data kind: The current sample rate

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.6 AddClock()

```
virtual AAX_Result AAX_IComponentDescriptor::AddClock (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a clock context field.

Defines a clock port for host-provided information in the algorithm's context structure.

- Data type: [AAX_CTimestamp](#) *
- Data kind: A running counter which increments even when the transport is not playing. The counter increments exactly once per sample quantum.

Host Compatibility Notes As of Pro Tools 11.1, this field may be used in both Native and DSP plug-ins. The DSP clock data is a 16-bit cycling counter. This field was only available for Native plug-ins in previous Pro Tools versions.

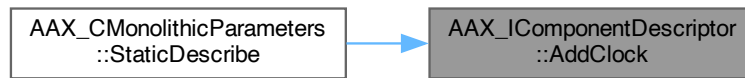
Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.7 AddSideChainIn()

```
virtual AAX_Result AAX_IComponentDescriptor::AddSideChainIn (
    AAX_CFieldIndex inFieldIndex ) [pure virtual]
```

Subscribes a side-chain input context field.

Defines a side-chain input port for host-provided information in the algorithm's context structure.

- Data type: `int32_t*`
- Data kind: The index of the plug-in's first side-chain input channel within the array of input audio buffers

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.8 AddDataInPort()

```
virtual AAX_Result AAX_IComponentDescriptor::AddDataInPort (
    AAX_CFieldIndex inFieldIndex,
    uint32_t inPacketSize,
    AAX_EDataInPortType inPortType = AAX_eDataInPortType_Buffered ) [pure virtual]
```

Adds a custom data port to the algorithm context.

Defines a read-only data port for plug-in information in the algorithm's context structure. The plug-in can send information to this port using [AAX_IController::PostPacket\(\)](#).

The host guarantees that all packets will be delivered to this port in the order in which they were posted, up to the point of a packet buffer overflow, though some packets may be dropped depending on the `inPortType` and host implementation.

Note

When a plug-in is operating in offline (AudioSuite) mode, all data ports operate as [AAX_eDataInPortType_Unbuffered](#) ports

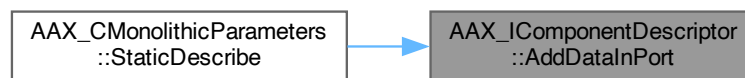
Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inPacketSize</i>	Size of the data packets that will be sent to this port
in	<i>inPortType</i>	The requested packet delivery behavior for this port

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.9 AddAuxOutputStem()

```
virtual AAX\_Result AAX_IComponentDescriptor::AddAuxOutputStem (
    AAX\_CFieldIndex inFieldIndex,
    int32_t inStemFormat,
    const char inNameUTF8[] ) [pure virtual]
```

Adds an auxiliary output stem for a plug-in.

Use this method to add additional output channels to the algorithm context.

The aux output stem audio buffers will be added to the end of the audio outputs array in the order in which they are described. When writing audio data to a specific aux output, find the proper starting channel by accumulating all of the channels of the main output stem format and any previously-described aux output stems.

The plug-in is responsible for providing a meaningful name for each aux outputs. At the very least, individual outputs should be labeled "Output xx", where "xx" is the aux output number as it is defined in the plug-in. The output name should also include the words "mono" and "stereo" to support when users are looking for an output with a specific stem format.

Host Compatibility Notes There is a hard limit to the number of outputs that Pro Tools supports for a single plug-in instance. This limit is currently set at 256 channels, which includes all of the plug-in's output channels in addition to the sum total of all of its aux output stem channels.

Host Compatibility Notes Pro Tools supports only mono and stereo auxiliary output stem formats

Warning

This method will return an error code on hosts which do not support auxiliary output stems. This indicates that the host will not provide audio buffers for auxiliary output stems during processing. A plug-in must not attempt to write data into auxiliary output stem buffers which have not been provided by the host!

Parameters

in	<i>inFieldIndex</i>	DEPRECATED: This parameter is no longer needed by the host, but is included in the interface for binary compatibility
in	<i>inStemFormat</i>	The stem format of the new aux output
in	<i>inNameUTF8</i>	The name of the aux output. This name is static and cannot be changed after the descriptor is submitted to the host

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.10 AddPrivateData()

```

virtual AAX\_Result AAX_IComponentDescriptor::AddPrivateData (
    AAX\_CFieldIndex inFieldIndex,
    int32_t inDataSize,
    uint32_t inOptions = AAX\_ePrivateDataOptions\_DefaultOptions ) [pure virtual]
  
```

Adds a private data port to the algorithm context.

Defines a read/write data port for private state data. Data written to this port will be maintained by the host between calls to the algorithm context.

See also

[alg_pd_registration](#)

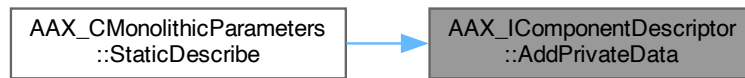
Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inDataSize</i>	Size of the data packets that will be sent to this port
in	<i>inOptions</i>	Options that define the private data port's behavior

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.11 AddTemporaryData()

```
virtual AAX_Result AAX_IComponentDescriptor::AddTemporaryData (
    AAX_CFieldIndex inFieldIndex,
    uint32_t inDataElementSize ) [pure virtual]
```

Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.

This can be very useful if you use block processing and need to store intermediate results. Just specify your base element size and the system will scale the overall block size by the buffer size. For example, to create a buffer of floats that is the length of the block, specify 4 bytes as the elementsize.

This data block does not retain state across callback and can also be reused across instances on memory constrained systems.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inDataElementSize</i>	The size of a single piece of data in the block. This number will be multiplied by the processing block size to determine total block size.

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.12 AddDmaInstance()

```
virtual AAX_Result AAX_IComponentDescriptor::AddDmaInstance (
    AAX_CFieldIndex inFieldIndex,
    AAX_IDma::EMode inDmaMode ) [pure virtual]
```

Adds a DMA field to the plug-in's context.

DMA (direct memory access) provides efficient reads from and writes to external memory on the DSP. DMA behavior is emulated in host-based plug-ins for cross-platform portability.

Note

The order in which DMA instances are added defines their priority and therefore order of execution of DMA operations. In most plug-ins, Scatter fields should be placed first in order to achieve the lowest possible access latency.

For more information, see [Direct Memory Access](#) .

Todo Update the DMA system management such that operation priority can be set arbitrarily

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inDmaMode</i>	AAX_IDma::EMode that will apply to this field

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.13 AddMeters()

```
virtual AAX\_Result AAX_IComponentDescriptor::AddMeters (
    AAX\_CFieldIndex inFieldIndex,
    const AAX\_CTypeID * inMeterIDs,
    const uint32_t inMeterCount ) [pure virtual]
```

Adds a meter field to the plug-in's context.

Meter fields include an array of meter tap values, with one tap per meter per context. Only one meter field should be added per Component. Individual meter behaviors can be described at the Effect level.

For more information, see [Plug-in meters](#) .

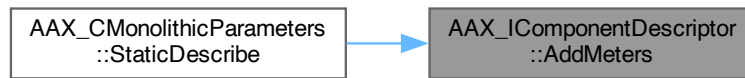
Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inMeterIDs</i>	Array of 32-bit IDs, one for each meter. Meter IDs must be unique within the Effect.
in	<i>inMeterCount</i>	The number of meters included in this field

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.14 AddMIDINode()

```

virtual AAX_Result AAX_IComponentDescriptor::AddMIDINode (
    AAX_CFieldIndex inFieldIndex,
    AAX_EMIDINodeType inNodeType,
    const char inNodeName[],
    uint32_t channelMask ) [pure virtual]
  
```

Adds a MIDI node field to the plug-in's context.

- Data type: [AAX_IMIDINode](#) *

The resulting MIDI node data will be available both in the algorithm context and in the plug-in's [data model](#) via [UpdateMIDINodes\(\)](#).

To add a MIDI node that is only accessible to the plug-in's data model, use [AAX_IEffectDescriptor::AddControlMIDINode\(\)](#)

Host Compatibility Notes Due to current restrictions MIDI data won't be delivered to DSP algorithms, only to AAX Native.

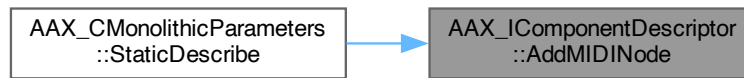
Parameters

in	<i>inFieldIndex</i>	The ID of the port. MIDI node ports should be formatted as a pointer to an AAX_IMIDINode .
in	<i>inNodeType</i>	The type of MIDI node, as AAX_EMIDINodeType
in	<i>inNodeName</i>	The name of the MIDI node as it should appear in the host's UI
in	<i>channelMask</i>	The channel mask for the MIDI node. This parameter specifies used MIDI channels. For Global MIDI nodes, use a mask of AAX_EMidGlobalNodeSelectors

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.15 AddReservedField()

```
virtual AAX_Result AAX_IComponentDescriptor::AddReservedField (
    AAX_CFieldIndex inFieldIndex,
    uint32_t inFieldType ) [pure virtual]
```

Subscribes a context field to host-provided services or information.

Note

Currently for internal use only.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inFieldType</i>	Type of field that is being added

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.16 NewPropertyMap()

```
virtual AAX_IPropertyMap * AAX_IComponentDescriptor::NewPropertyMap ( ) const [pure virtual]
```

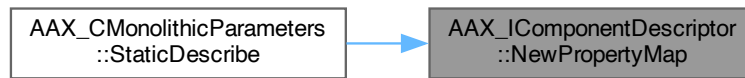
Creates a new, empty property map.

The component descriptor owns the reference to the resulting property map, and the underlying property map is destroyed when the component descriptor is released.

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.17 DuplicatePropertyMap()

```
virtual AAX_IPropertyMap * AAX_IComponentDescriptor::DuplicatePropertyMap (
    AAX_IPropertyMap * inPropertyMap ) const [pure virtual]
```

Creates a new property map using an existing property map.

The component descriptor owns the reference to the resulting property map, and the underlying property map is destroyed when the component descriptor is released.

Parameters

in	<i>inPropertyMap</i>	The property values in this map will be copied into the new map
----	----------------------	---

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.18 AddProcessProc_Native() [1/2]

```
virtual AAX_Result AAX_IComponentDescriptor::AddProcessProc_Native (
    AAX_CProcessProc inProcessProc,
    AAX_IPropertyMap * inProperties = NULL,
    AAX_CInstanceInitProc inInstanceInitProc = NULL,
    AAX_CBackgroundProc inBackgroundProc = NULL,
    AAX_CSelector * outProcID = NULL ) [pure virtual]
```

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

Parameters

in	<i>inProcessProc</i>	Symbol for this processing callback
in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
in	<i>inInstanceInitProc</i>	Initialization routine that will be called when a new instance of the Effect is created. See Algorithm initialization .

Parameters

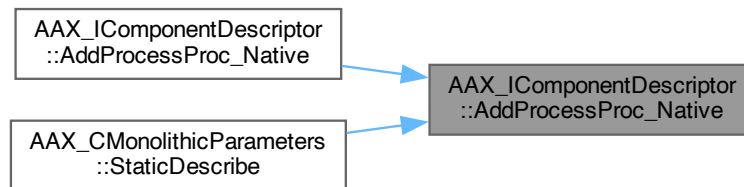
in	<i>inBackgroundProc</i>	Background routine that will be called in an idle context within the same address space as the associated process procedure. See Background processing callback
out	<i>outProcID</i>	

Todo document this parameter

Implemented in [AAX_VComponentDescriptor](#).

Referenced by [AddProcessProc_Native\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.98.3.19 AddProcessProc_TI()

```

virtual AAX\_Result AAX_IComponentDescriptor::AddProcessProc_TI (
    const char inDLLFileNameUTF8[],
    const char inProcessProcSymbol[],
    AAX\_IPropertyMap * inProperties,
    const char inInstanceInitProcSymbol[] = NULL,
    const char inBackgroundProcSymbol[] = NULL,
    AAX\_CSelector * outProcID = NULL ) [pure virtual]
  
```

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

Parameters

in	<i>inDLLFileNameUTF8</i>	UTF-8 encoded filename for the ELF DLL containing the algorithm code fragment
in	<i>inProcessProcSymbol</i>	Symbol for this processing callback
in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.

Parameters

in	<i>inInstanceInitProcSymbol</i>	Initialization routine that will be called when a new instance of the Effect is created. Must be included in the same DLL as the main algorithm entrypoint. See Algorithm initialization .
in	<i>inBackgroundProcSymbol</i>	Background routine that will be called in an idle context within the same address space as the associated process procedure. Must be included in the same DLL as the main algorithm entrypoint. See Background processing callback
out	<i>outProcID</i>	

Todo document this parameter

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.20 AddProcessProc()

```
virtual AAX_Result AAX_IComponentDescriptor::AddProcessProc (
    AAX_IPropertyMap * inProperties,
    AAX_CSelector * outProcIDs = NULL,
    int32_t inProcIDsSize = 0 ) [pure virtual]
```

Registers one or more algorithm processing entrypoints (process procedures)

Any non-overlapping set of processing entrypoints may be specified. Typically this can be used to specify both Native and TI entrypoints using the same call.

The AAX Library implementation of this method includes backwards compatibility logic to complete the ProcessProc registration on hosts which do not support this method. Therefore plug-in code may use this single registration routine instead of separate calls to [AddProcessProc_Native\(\)](#), [AddProcessProc_TI\(\)](#), etc. regardless of the host version.

The following properties replace the input arguments to the platform-specific registration methods:

[AddProcessProc_Native\(\)](#) ([AAX_eProperty_PluginID_Native](#), [AAX_eProperty_PluginID_AudioSuite](#))

- [AAX_CProcessProc](#) iProcessProc: [AAX_eProperty_NativeProcessProc](#) (required)
- [AAX_CInstanceInitProc](#) iInstanceInitProc: [AAX_eProperty_NativeInstanceInitProc](#) (optional)
- [AAX_CBackgroundProc](#) iBackgroundProc: [AAX_eProperty_NativeBackgroundProc](#) (optional)

[AddProcessProc_TI\(\)](#) ([AAX_eProperty_PluginID_TI](#))

- `const char inDLLFileNameUTF8[]`: [AAX_eProperty_TIDLLFileName](#) (required)
- `const char iProcessProcSymbol[]`: [AAX_eProperty_TIPProcessProc](#) (required)
- `const char iInstanceInitProcSymbol[]`: [AAX_eProperty_TIInstanceInitProc](#) (optional)
- `const char iBackgroundProcSymbol[]`: [AAX_eProperty_TIBackgroundProc](#) (optional)

If any platform-specific plug-in ID property is present in `iProperties` then [AddProcessProc\(\)](#) will check for the required properties for that platform.

Note

[AAX_eProperty_AudioBufferLength](#) will be ignored for the Native and AudioSuite ProcessProcs since it should only be used for AAX DSP.

Parameters

in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
out	<i>outProcIDs</i>	

Todo document this parameter Returned array will be NULL-terminated

Parameters

in	<i>inProcIDsSize</i>	The size of the array provided to oProcIDs. If oProcIDs is non-NULL but iProcIDsSize is not large enough for all of the registered ProcessProcs (plus one for NULL termination) then this method will fail with AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW
----	----------------------	--

Implemented in [AAX_VComponentDescriptor](#).

14.98.3.21 AddProcessProc_Native() [2/2]

```
template<typename aContextType >
AAX_Result AAX_IComponentDescriptor::AddProcessProc_Native (
    void(AAX_CALLBACK *inProcessProc)(aContextType *const inInstancesBegin[], const
void *inInstancesEnd) ,
    AAX_IPropertyMap * inProperties = NULL,
    int32_t(AAX_CALLBACK *inInstanceInitProc)(const aContextType *inInstanceContext←
Ptr, AAX_EComponentInstanceInitAction inAction) = NULL,
    int32_t(AAX_CALLBACK *inBackgroundProc)(void) = NULL ) [inline]
```

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

This template provides an [AAX_CALLBACK](#) based interface to the [AddProcessProc_Native](#) method.

See also

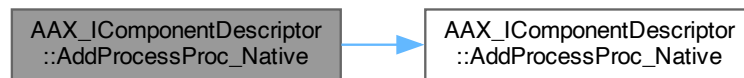
[AAX_IComponentDescriptor::AddProcessProc_Native\(AAX_CProcessProc,AAX_IPropertyMap*,AAX_CInstanceInitProc,AAX_](#)

Parameters

in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
----	---------------------	---

References [AddProcessProc_Native\(\)](#).

Here is the call graph for this function:



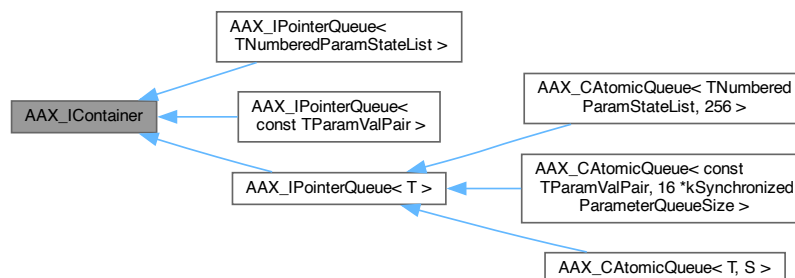
The documentation for this class was generated from the following file:

- [AAX_IComponentDescriptor.h](#)

14.99 AAX_IContainer Class Reference

```
#include <AAX_IContainer.h>
```

Inheritance diagram for AAX_IContainer:



14.99.1 Description

Abstract container interface

Public Types

- enum [EStatus](#) {
[eStatus_Success](#) = 0 ,
[eStatus_Overflow](#) = 1 ,
[eStatus_NotInitialized](#) = 2 ,
[eStatus_Unavailable](#) = 3 ,
[eStatus_Unsupported](#) = 4 }

Public Member Functions

- virtual [~AAX_IContainer](#) ()
- virtual void [Clear](#) ()=0

14.99.2 Member Enumeration Documentation

14.99.2.1 EStatus

enum [AAX_IContainer::EStatus](#)

Enumerator

eStatus_Success	Operation succeeded.
eStatus_Overflow	Internal buffer overflow.
eStatus_NotInitialized	Uninitialized container.
eStatus_Unavailable	An internal resource was not available.
eStatus_Unsupported	Operation is unsupported.

14.99.3 Constructor & Destructor Documentation

14.99.3.1 ~AAX_IContainer()

```
virtual AAX_IContainer::~~AAX_IContainer ( ) [inline], [virtual]
```

14.99.4 Member Function Documentation

14.99.4.1 Clear()

```
virtual void AAX_IContainer::Clear ( ) [pure virtual]
```

Clear the container

Implemented in [AAX_CAtomicQueue< T, S >](#), [AAX_CAtomicQueue< TNumberedParamStateList, 256 >](#), [AAX_CAtomicQueue< const TParamValPair, 16 *kSynchronizedParameterQueueSize >](#), [AAX_IPointerQueue< T >](#), [AAX_IPointerQueue< TNumberedParamStateList >](#), and [AAX_IPointerQueue< const TParamValPair >](#).

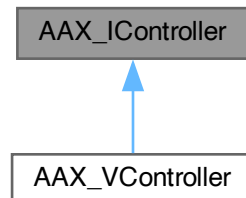
The documentation for this class was generated from the following file:

- [AAX_IContainer.h](#)

14.100 AAX_IController Class Reference

```
#include <AAX_IController.h>
```

Inheritance diagram for AAX_IController:



14.100.1 Description

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAX host and by effect components.

:Implemented by the AAX Host

Public Member Functions

- virtual [~AAX_IController](#) (void)

Host information getters

Call these methods to retrieve environment and run-time information from the AAX host.

- virtual [AAX_Result GetEffectID](#) ([AAX_IString](#) *outEffectID) const =0
- virtual [AAX_Result GetSampleRate](#) ([AAX_CSampleRate](#) *outSampleRate) const =0
CALL: Returns the current literal sample rate.
- virtual [AAX_Result GetInputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's input stem format.
- virtual [AAX_Result GetOutputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const =0
CALL: Returns the plug-in's output stem format.
- virtual [AAX_Result GetSignalLatency](#) (int32_t *outSamples) const =0
CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.
- virtual [AAX_Result GetCycleCount](#) ([AAX_EProperty](#) inWhichCycleCount, [AAX_CPropertyValue](#) *out← NumCycles) const =0
CALL: returns the plug-in's current real-time DSP cycle count.
- virtual [AAX_Result GetTODLocation](#) ([AAX_CTimeOfDay](#) *outTODLocation) const =0
CALL: Returns the current Time Of Day (TOD) of the system.

Host information setters

Call these methods to set dynamic plug-in run-time information on the AAX host.

- virtual [AAX_Result SetSignalLatency](#) (int32_t inNumSamples)=0
CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.
- virtual [AAX_Result SetCycleCount](#) (AAX_EProperty *inWhichCycleCounts, AAX_CPropertyValue *inValues, int32_t numValues)=0
CALL: Indicates a change in the plug-in's real-time DSP cycle count.

Posting methods

Call these methods to post new plug-in information to the host's data management system.

- virtual [AAX_Result PostPacket](#) (AAX_CFieldIndex inFieldIndex, const void *inPayloadP, uint32_t inPayloadSize)=0
CALL: Posts a data packet to the host for routing between plug-in components.

Notification methods

Call these methods to send events among plug-in components

- virtual [AAX_Result SendNotification](#) (AAX_CTypeID inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
CALL: Dispatch a notification.
- virtual [AAX_Result SendNotification](#) (AAX_CTypeID inNotificationType)=0
CALL: Sends an event to the GUI (no payload)

Metering methods

Methods to access the plug-in's host-managed metering information.

See also

[Plug-in meters](#)

- virtual [AAX_Result GetCurrentMeterValue](#) (AAX_CTypeID inMeterID, float *outMeterValue) const =0
CALL: Retrieves the current value of a host-managed plug-in meter.
- virtual [AAX_Result GetMeterPeakValue](#) (AAX_CTypeID inMeterID, float *outMeterPeakValue) const =0
CALL: Retrieves the currently held peak value of a host-managed plug-in meter.
- virtual [AAX_Result ClearMeterPeakValue](#) (AAX_CTypeID inMeterID) const =0
CALL: Clears the peak value from a host-managed plug-in meter.
- virtual [AAX_Result GetMeterCount](#) (uint32_t *outMeterCount) const =0
CALL: Retrieves the number of host-managed meters registered by a plug-in.
- virtual [AAX_Result GetMeterClipped](#) (AAX_CTypeID inMeterID, AAX_CBoolean *outClipped) const =0
CALL: Retrieves the clipped flag from a host-managed plug-in meter.
- virtual [AAX_Result ClearMeterClipped](#) (AAX_CTypeID inMeterID) const =0
CALL: Clears the clipped flag from a host-managed plug-in meter.

MIDI methods

Methods to access the plug-in's host-managed MIDI information.

- virtual [AAX_Result GetNextMIDIPacket](#) (AAX_CFieldIndex *outPort, AAX_CMidiPacket *outPacket)=0
CALL: Retrieves MIDI packets for described MIDI nodes.
- virtual [AAX_Result GetHybridSignalLatency](#) (int32_t *outSamples) const =0
CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.
- virtual [AAX_Result GetCurrentAutomationTimestamp](#) (AAX_CTransportCounter *outTimestamp) const =0
CALL: Returns the current automation timestamp if called during the [GenerateCoefficients\(\)](#) call AND the generation of coefficients is being triggered by an automation point instead of immediate changes.
- virtual [AAX_Result GetHostName](#) (AAX_IString *outHostNameString) const =0
CALL: Returns name of the host application this plug-in instance is being loaded by. This string also typically includes version information.

- virtual [AAX_Result](#) [GetPlugInTargetPlatform](#) ([AAX_CTargetPlatform](#) *outTargetPlatform) const =0
CALL: Returns execution platform type, native or TI.
- virtual [AAX_Result](#) [GetIsAudioSuite](#) ([AAX_CBoolean](#) *outIsAudioSuite) const =0
CALL: Returns true for AudioSuite instances.
- virtual [AAX_IPageTable](#) * [CreateTableCopyForEffect](#) ([AAX_CPropertyValue](#) inManufacturerID, [AAX_CPropertyValue](#) inProductID, [AAX_CPropertyValue](#) inPlugInID, uint32_t inTableType, int32_t inTablePageSize) const =0
Copy the current page table data for a particular plug-in type.
- virtual [AAX_IPageTable](#) * [CreateTableCopyForLayout](#) (const char *inEffectID, const char *inLayoutName, uint32_t inTableType, int32_t inTablePageSize) const =0
Copy the current page table data for a particular plug-in effect and page table layout.
- virtual [AAX_IPageTable](#) * [CreateTableCopyForEffectFromFile](#) (const char *inPageTableFilePath, [AAX_ETextEncoding](#) inFilePathEncoding, [AAX_CPropertyValue](#) inManufacturerID, [AAX_CPropertyValue](#) inProductID, [AAX_CPropertyValue](#) inPlugInID, uint32_t inTableType, int32_t inTablePageSize) const =0
Copy the current page table data for a particular plug-in type.
- virtual [AAX_IPageTable](#) * [CreateTableCopyForLayoutFromFile](#) (const char *inPageTableFilePath, [AAX_ETextEncoding](#) inFilePathEncoding, const char *inLayoutName, uint32_t inTableType, int32_t inTablePageSize) const =0
Copy the current page table data for a particular plug-in effect and page table layout.

14.100.2 Constructor & Destructor Documentation

14.100.2.1 ~AAX_IController()

```
virtual AAX_IController::~~AAX_IController (
    void ) [inline], [virtual]
```

14.100.3 Member Function Documentation

14.100.3.1 GetEffectID()

```
virtual AAX\_Result AAX_IController::GetEffectID (
    AAX\_IString * outEffectID ) const [pure virtual]
```

Implemented in [AAX_VController](#).

14.100.3.2 GetSampleRate()

```
virtual AAX\_Result AAX_IController::GetSampleRate (
    AAX\_CSampleRate * outSampleRate ) const [pure virtual]
```

CALL: Returns the current literal sample rate.

Parameters

out	<i>outSampleRate</i>	The current sample rate
-----	----------------------	-------------------------

Implemented in [AAX_VController](#).

14.100.3.3 GetInputStemFormat()

```
virtual AAX_Result AAX_IController::GetInputStemFormat (
    AAX_EStemFormat * outStemFormat ) const [pure virtual]
```

CALL: Returns the plug-in's input stem format.

Parameters

out	<i>outStemFormat</i>	The current input stem format
-----	----------------------	-------------------------------

Implemented in [AAX_VController](#).

14.100.3.4 GetOutputStemFormat()

```
virtual AAX_Result AAX_IController::GetOutputStemFormat (
    AAX_EStemFormat * outStemFormat ) const [pure virtual]
```

CALL: Returns the plug-in's output stem format.

Parameters

out	<i>outStemFormat</i>	The current output stem format
-----	----------------------	--------------------------------

Implemented in [AAX_VController](#).

14.100.3.5 GetSignalLatency()

```
virtual AAX_Result AAX_IController::GetSignalLatency (
    int32_t * outSamples ) const [pure virtual]
```

CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.

This method provides the most recently published signal latency. The host may not have updated its delay compensation to match this signal latency yet, so plug-ins that dynamically change their latency using [SetSignalLatency\(\)](#) should always wait for an [AAX_eNotificationEvent_SignalLatencyChanged](#) notification before updating its algorithm to incur this latency.

See also

[SetSignalLatency\(\)](#)

Parameters

out	<i>outSamples</i>	The number of samples of signal delay published by the plug-in
-----	-------------------	--

Implemented in [AAX_VController](#).

14.100.3.6 GetCycleCount()

```
virtual AAX_Result AAX_IController::GetCycleCount (
    AAX_EProperty inWhichCycleCount,
    AAX_CPropertyValue * outNumCycles ) const [pure virtual]
```

CALL: returns the plug-in's current real-time DSP cycle count.

This method provides the number of cycles that the AAX host expects the DSP plug-in to consume. The host uses this value when allocating DSP resources for the plug-in.

Note

A plug-in should never apply a DSP algorithm with more demanding resource requirements than what is currently accounted for by the host. To set a higher cycle count value, a plug-in must call [AAX_IController::SetCycleCount\(\)](#), then poll [AAX_IController::GetCycleCount\(\)](#) until the new value has been applied. Once the host has recognized the new cycle count value, the plug-in may apply the more demanding algorithm.

Parameters

in	<i>inWhichCycleCount</i>	Selector for the requested cycle count metric. One of: <ul style="list-style-type: none"> • AAX_eProperty_TI_SharedCycleCount • AAX_eProperty_TI_InstanceCycleCount • AAX_eProperty_TI_MaxInstancesPerChip
in	<i>outNumCycles</i>	The current value of the selected cycle count metric

Todo PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

Implemented in [AAX_VController](#).

14.100.3.7 GetTODLocation()

```
virtual AAX_Result AAX_IController::GetTODLocation (
    AAX_CTimeOfDay * outTODLocation ) const [pure virtual]
```

CALL: Returns the current Time Of Day (TOD) of the system.

This method provides a plug-in the TOD (in samples) of the current system. TOD is the number of samples that the playhead has traversed since the beginning of playback.

Note

The TOD value is the immediate value of the audio engine playhead. This value is incremented within the audio engine's real-time rendering context; it is not synchronized with non-real-time calls to plug-in interface methods.

Parameters

out	<i>outTODLocation</i>	The current Time Of Day as set by the host
-----	-----------------------	--

Implemented in [AAX_VController](#).

14.100.3.8 SetSignalLatency()

```
virtual AAX_Result AAX_IController::SetSignalLatency (
    int32_t inNumSamples ) [pure virtual]
```

CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.

This method is used to request a change in the number of samples that the AAX host expects the plug-in to delay a signal.

The host is not guaranteed to immediately apply the new latency value. A plug-in should avoid incurring an actual algorithmic latency that is different than the latency accounted for by the host.

To set a new latency value, a plug-in must call [AAX_IController::SetSignalLatency\(\)](#), then wait for an [AAX_eNotificationEvent_SignalLatencyChanged](#) notification. Once this notification has been received, [AAX_IController::GetSignalLatency\(\)](#) will reflect the updated latency value and the plug-in should immediately apply any relevant algorithmic changes that alter its latency to this new value.

Warning

Parameters which affect the latency of a plug-in should not be made available for control through automation. This will result in audible glitches when delay compensation is adjusted while playing back automation for these parameters.

Parameters

in	<i>inNumSamples</i>	The number of samples of signal delay that the plug-in requests to incur
----	---------------------	--

Implemented in [AAX_VController](#).

14.100.3.9 SetCycleCount()

```
virtual AAX_Result AAX_IController::SetCycleCount (
    AAX_EProperty * inWhichCycleCounts,
    AAX_CPropertyValue * iValues,
    int32_t numValues ) [pure virtual]
```

CALL: Indicates a change in the plug-in's real-time DSP cycle count.

This method is used to request a change in the number of cycles that the AAX host expects the DSP plug-in to consume.

Note

A plug-in should never apply a DSP algorithm with more demanding resource requirements than what is currently accounted for by the host. To set a higher cycle count value, a plug-in must call [AAX_IController::SetCycleCount\(\)](#), then poll [AAX_IController::GetCycleCount\(\)](#) until the new value has been applied. Once the host has recognized the new cycle count value, the plug-in may apply the more demanding algorithm.

Parameters

in	<i>inWhichCycleCounts</i>	Array of selectors indicating the specific cycle count metrics that should be set. Each selector must be one of: <ul style="list-style-type: none"> • AAX_eProperty_TI_SharedCycleCount • AAX_eProperty_TI_InstanceCycleCount • AAX_eProperty_TI_MaxInstancesPerChip
in	<i>iValues</i>	An array of values requested, one for each of the selected cycle count metrics.
in	<i>numValues</i>	The size of iValues

Todo PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

Implemented in [AAX_VController](#).

14.100.3.10 PostPacket()

```
virtual AAX_Result AAX_IController::PostPacket (
    AAX_CFieldIndex inFieldIndex,
    const void * inPayloadP,
    uint32_t inPayloadSize ) [pure virtual]
```

CALL: Posts a data packet to the host for routing between plug-in components.

The posted packet is identified with a [AAX_CFieldIndex](#) packet index value, which is equivalent to the target data port's identifier. The packet's payload must have the expected size for the given packet index / data port, as defined when the port is created in [Describe](#). See [AAX_IComponentDescriptor::AddDataInPort\(\)](#).

Warning

Any data structures that will be passed between platforms (for example, sent to a TI DSP in an AAX DSP plug-in) must be properly data-aligned for compatibility across both platforms. See [AAX_ALIGN_FILE_ALG](#) for more information about guaranteeing cross-platform compatibility of data structures used for algorithm processing.

Note

All calls to this method should be made within the scope of [AAX_IEffectParameters::GenerateCoefficients\(\)](#). Calls from outside this method may result in packets not being delivered. See [PT-206161](#)

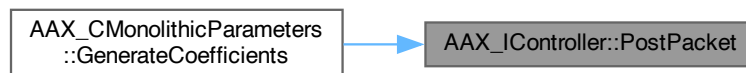
Parameters

in	<i>inFieldIndex</i>	The packet's destination port
in	<i>inPayloadP</i>	A pointer to the packet's payload data
in	<i>inPayloadSize</i>	The size, in bytes, of the payload data

Implemented in [AAX_VController](#).

Referenced by [AAX_CMonolithicParameters::GenerateCoefficients\(\)](#).

Here is the caller graph for this function:

**14.100.3.11 SendNotification() [1/2]**

```

virtual AAX\_Result AAX_IController::SendNotification (
    AAX\_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [pure virtual]
  
```

CALL: Dispatch a notification.

The notification is handled by the host and may be delivered back to other plug-in components such as the GUI or data model (via [AAX_IEffectGUI::NotificationReceived\(\)](#) or [AAX_IEffectParameters::NotificationReceived\(\)](#), respectively) depending on the notification type.

The host may choose to dispatch the posted notification either synchronously or asynchronously.

See the [AAX_ENotificationEvent](#) documentation for more information.

This method is supported by AAX V2 Hosts only. Check the return code on the return of this function. If the error is [AAX_ERROR_UNIMPLEMENTED](#), your plug-in is being loaded into a host that doesn't support this feature.

Parameters

in	<i>inNotificationType</i>	Type of notification to send
in	<i>inNotificationData</i>	Block of notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implemented in [AAX_VController](#).

14.100.3.12 SendNotification() [2/2]

```
virtual AAX_Result AAX_IController::SendNotification (
    AAX_CTypeID inNotificationType ) [pure virtual]
```

CALL: Sends an event to the GUI (no payload)

This version of the notification method is a convenience for notifications which do not take any payload data. Internally, it simply calls [AAX_IController::SendNotification\(AAX_CTypeID, const void*, uint32_t\)](#) with a null payload.

Parameters

in	<i>inNotificationType</i>	Type of notification to send
----	---------------------------	------------------------------

Implemented in [AAX_VController](#).

14.100.3.13 GetCurrentMeterValue()

```
virtual AAX_Result AAX_IController::GetCurrentMeterValue (
    AAX_CTypeID inMeterID,
    float * outMeterValue ) const [pure virtual]
```

CALL: Retrieves the current value of a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried
out	<i>outMeterValue</i>	The queried meter's current value

Implemented in [AAX_VController](#).

14.100.3.14 GetMeterPeakValue()

```
virtual AAX_Result AAX_IController::GetMeterPeakValue (
    AAX_CTypeID inMeterID,
    float * outMeterPeakValue ) const [pure virtual]
```

CALL: Retrieves the currently held peak value of a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried
out	<i>outMeterPeakValue</i>	The queried meter's currently held peak value

Implemented in [AAX_VController](#).

14.100.3.15 ClearMeterPeakValue()

```
virtual AAX_Result AAX_IController::ClearMeterPeakValue (
    AAX_CTypeID inMeterID ) const [pure virtual]
```

CALL: Clears the peak value from a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being cleared
----	------------------	---------------------------------------

Implemented in [AAX_VController](#).

14.100.3.16 GetMeterCount()

```
virtual AAX_Result AAX_IController::GetMeterCount (
    uint32_t * outMeterCount ) const [pure virtual]
```

CALL: Retrieves the number of host-managed meters registered by a plug-in.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

out	<i>outMeterCount</i>	The number of registered plug-in meters.
-----	----------------------	--

Implemented in [AAX_VController](#).

14.100.3.17 GetMeterClipped()

```
virtual AAX_Result AAX_IController::GetMeterClipped (
    AAX_CTypeID inMeterID,
    AAX_CBoolean * outClipped ) const [pure virtual]
```

CALL: Retrieves the clipped flag from a host-managed plug-in meter.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried.
out	<i>outClipped</i>	The queried meter's clipped flag.

Implemented in [AAX_VController](#).

14.100.3.18 ClearMeterClipped()

```
virtual AAX_Result AAX_IController::ClearMeterClipped (
    AAX_CTypeID inMeterID ) const [pure virtual]
```

CALL: Clears the clipped flag from a host-managed plug-in meter.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

in	<i>inMeterID</i>	ID of the meter that is being cleared.
----	------------------	--

Implemented in [AAX_VController](#).

14.100.3.19 GetNextMIDIPacket()

```
virtual AAX_Result AAX_IController::GetNextMIDIPacket (
    AAX_CFieldIndex * outPort,
    AAX_CMidiPacket * outPacket ) [pure virtual]
```

CALL: Retrieves MIDI packets for described MIDI nodes.

Parameters

out	<i>outPort</i>	port ID of the MIDI node that has unhandled packet
out	<i>outPacket</i>	The MIDI packet

Implemented in [AAX_VController](#).

14.100.3.20 GetCurrentAutomationTimestamp()

```
virtual AAX_Result AAX_IController::GetCurrentAutomationTimestamp (
    AAX_CTransportCounter * outTimestamp ) const [pure virtual]
```

CALL: Returns the current automation timestamp if called during the [GenerateCoefficients\(\)](#) call AND the generation of coefficients is being triggered by an automation point instead of immediate changes.

Note

This function will return 0 if called from outside of [GenerateCoefficients\(\)](#) or if the [GenerateCoefficients\(\)](#) call was initiated due to a non-automated change. In those cases, you can get your sample offset from the transport start using [GetTODLocation\(\)](#).

Parameters

out	<i>outTimestamp</i>	The current coefficient timestamp. Sample count from transport start.
-----	---------------------	---

Implemented in [AAX_VController](#).

14.100.3.21 GetHostName()

```
virtual AAX_Result AAX_IController::GetHostName (
    AAX_IString * outHostNameString ) const [pure virtual]
```

CALL: Returns name of the host application this plug-in instance is being loaded by. This string also typically includes version information.

Host Compatibility Notes Pro Tools versions from Pro Tools 11.0 to Pro Tools 12.3.1 will return a generic version string to this call. This issue is resolved beginning in Pro Tools 12.4.

Parameters

out	<i>outHostNameString</i>	The name of the current host application.
-----	--------------------------	---

Implemented in [AAX_VController](#).

14.100.3.22 GetPlugInTargetPlatform()

```
virtual AAX_Result AAX_IController::GetPlugInTargetPlatform (
    AAX_CTargetPlatform * outTargetPlatform ) const [pure virtual]
```

CALL: Returns execution platform type, native or TI.

Parameters

out	<i>outTargetPlatform</i>	The type of the current execution platform as one of AAX_ETargetPlatform .
-----	--------------------------	--

Implemented in [AAX_VController](#).

14.100.3.23 GetIsAudioSuite()

```
virtual AAX_Result AAX_IController::GetIsAudioSuite (
    AAX_CBoolean * outIsAudioSuite ) const [pure virtual]
```

CALL: Returns true for AudioSuite instances.

Parameters

out	<i>outIsAudioSuite</i>	The boolean flag which indicate true for AudioSuite instances.
-----	------------------------	--

Implemented in [AAX_VController](#).

14.100.3.24 CreateTableCopyForEffect()

```
virtual AAX_IPageTable * AAX_IController::CreateTableCopyForEffect (
    AAX_CPropertyValue inManufacturerID,
    AAX_CPropertyValue inProductID,
    AAX_CPropertyValue inPlugInID,
    uint32_t inTableType,
    int32_t inTablePageSize ) const [pure virtual]
```

Copy the current page table data for a particular plug-in type.

The host may restrict plug-ins to only copying page table data from certain plug-in types, such as plug-ins from the same manufacturer or plug-in types within the same effect.

See [Page Table Guide](#) for more information about page tables.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if the requested plug-in type is unknown, if *inTableType* is unknown or if *inTablePageSize* is not a supported size for the given table type.

Parameters

in	<i>inManufacturerID</i>	Manufacturer ID of the desired plug-in type
in	<i>inProductID</i>	Product ID of the desired plug-in type
in	<i>inPlugInID</i>	Type ID of the desired plug-in type (AAX_eProperty_PlugInID_Native , AAX_eProperty_PlugInID_TI)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implemented in [AAX_VController](#).

14.100.3.25 CreateTableCopyForLayout()

```
virtual AAX_IPageTable * AAX_IController::CreateTableCopyForLayout (
    const char * inEffectID,
    const char * inLayoutName,
    uint32_t inTableType,
    int32_t inTablePageSize ) const [pure virtual]
```

Copy the current page table data for a particular plug-in effect and page table layout.

The host may restrict plug-ins to only copying page table data from certain effects, such as effects registered within the current [AAX](#) plug-in bundle.

See [Page Table Guide](#) for more information about page tables.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if the requested effect ID is unknown or if `inLayoutName` is not a valid layout name for the page tables registered for the effect.

Parameters

in	<i>inEffectID</i>	Effect ID for the desired effect. See AAX_ICollection::AddEffect()
in	<i>inLayoutName</i>	Page table layout name ("name" attribute of the <code>PTLayout</code> XML tag)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implemented in [AAX_VController](#).

14.100.3.26 CreateTableCopyForEffectFromFile()

```
virtual AAX_IPageTable * AAX_IController::CreateTableCopyForEffectFromFile (
    const char * inPageTableFilePath,
    AAX_ETextEncoding inFilePathEncoding,
    AAX_CPropertyValue inManufacturerID,
    AAX_CPropertyValue inProductID,
    AAX_CPropertyValue inPlugInID,
    uint32_t inTableType,
    int32_t inTablePageSize ) const [pure virtual]
```

Copy the current page table data for a particular plug-in type.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if the requested plug-in type is unknown, if `inTableType` is unknown or if `inTablePageSize` is not a supported size for the given table type.

Parameters

in	<i>inPageTableFilePath</i>	Path to XML page table file.
in	<i>inFilePathEncoding</i>	File path text encoding.
in	<i>inManufacturerID</i>	Manufacturer ID of the desired plug-in type
in	<i>inProductID</i>	Product ID of the desired plug-in type
in	<i>inPlugInID</i>	Type ID of the desired plug-in type (AAX_eProperty_PlugInID_Native , AAX_eProperty_PlugInID_TI)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implemented in [AAX_VController](#).

14.100.3.27 CreateTableCopyForLayoutFromFile()

```
virtual AAX\_IPageTable * AAX_IController::CreateTableCopyForLayoutFromFile (
    const char * inPageTableFilePath,
    AAX\_ETextEncoding inFilePathEncoding,
    const char * inLayoutName,
    uint32_t inTableType,
    int32_t inTablePageSize ) const [pure virtual]
```

Copy the current page table data for a particular plug-in effect and page table layout.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if *inLayoutName* is not a valid layout name for the page tables file.

Parameters

in	<i>inPageTableFilePath</i>	Path to XML page table file.
in	<i>inFilePathEncoding</i>	File path text encoding.
in	<i>inLayoutName</i>	Page table layout name ("name" attribute of the <code>PTLayout</code> XML tag)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implemented in [AAX_VController](#).

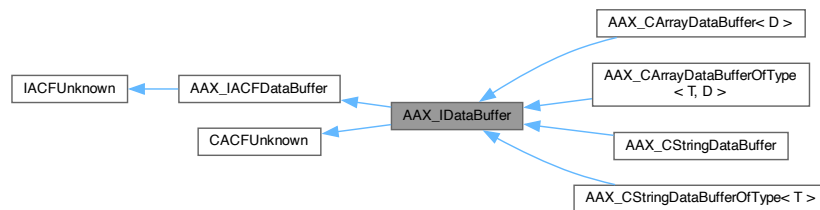
The documentation for this class was generated from the following file:

- [AAX_IController.h](#)

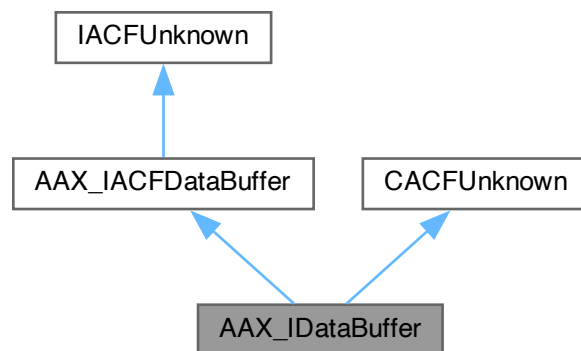
14.101 AAX_IDataBuffer Class Reference

```
#include <AAX_IDataBuffer.h>
```

Inheritance diagram for AAX_IDataBuffer:



Collaboration diagram for AAX_IDataBuffer:



14.101.1 Description

Interface for reference counted data buffers.

This interface is intended to be used for passing arbitrary blocks of data across the binary boundary and allowing the receiver to take ownership of the allocated memory.

Public Member Functions

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acflID](#) &riid
- [AAX_DELETE](#) (AAX_IDataBuffer &operator=(const [AAX_IDataBuffer](#) &))

Public Member Functions inherited from [AAX_IACFDataBuffer](#)

- virtual [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result Size](#) (int32_t *oSize) const =0
- virtual [AAX_Result Data](#) (void const **oBuffer) const =0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Public Attributes

- void **ppvObjOut [AAX_OVERRIDE](#)

14.101.2 Member Function Documentation**14.101.2.1 ACF_DECLARE_STANDARD_UNKNOWN()**

```
AAX_IDataBuffer::ACF_DECLARE_STANDARD_UNKNOWN ( ) const &
```

14.101.2.2 AAX_DELETE()

```
AAX_IDataBuffer::AAX_DELETE (
    AAX\_IDataBuffer & operator = (const AAX\_IDataBuffer &) )
```

14.101.3 Member Data Documentation

14.101.3.1 AAX_OVERRIDE

```
void** ppvObjOut AAX_IDataBuffer::AAX_OVERRIDE
```

Initial value:

```
{
    if (riid == IID_IAAXDataBufferV1)
    {
        *ppvObjOut = static_cast<IACFUnknown*>(this);
        ( static_cast<IACFUnknown*>(*ppvObjOut) )->AddRef();
        return ACF_OK;
    }

    return this->CACFUnknown::InternalQueryInterface(riid, ppvObjOut)
```

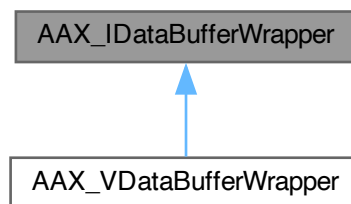
The documentation for this class was generated from the following file:

- [AAX_IDataBuffer.h](#)

14.102 AAX_IDataBufferWrapper Class Reference

```
#include <AAX_IDataBufferWrapper.h>
```

Inheritance diagram for AAX_IDataBufferWrapper:



14.102.1 Description

Wrapper for an [AAX_IDataBuffer](#).

Like [AAX_IController](#) and similar classes, this class provides a non-ACF interface matching an ACF interface, in this case [AAX_IACFDataBuffer](#) .

The implementation of this interface will contain a reference counted pointer to the underlying ACF interface. This interface may be extended with convenience functions that are not required on the underlying ACF interface.

Public Member Functions

- virtual [~AAX_IDataBufferWrapper](#) ()=default
- virtual [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result Size](#) (int32_t *oSize) const =0
- virtual [AAX_Result Data](#) (void const **oBuffer) const =0

14.102.2 Constructor & Destructor Documentation

14.102.2.1 ~AAX_IDataBufferWrapper()

```
virtual AAX_IDataBufferWrapper::~~AAX_IDataBufferWrapper ( ) [virtual], [default]
```

14.102.3 Member Function Documentation

14.102.3.1 Type()

```
virtual AAX_Result AAX_IDataBufferWrapper::Type (
    AAX_CTypeID * oType ) const [pure virtual]
```

The type of data contained in this buffer

This identifier must be sufficient for a client that knows the type to correctly interpret and use the data.

Implemented in [AAX_VDataBufferWrapper](#).

14.102.3.2 Size()

```
virtual AAX_Result AAX_IDataBufferWrapper::Size (
    int32_t * oSize ) const [pure virtual]
```

The number of bytes of data in this buffer

Implemented in [AAX_VDataBufferWrapper](#).

14.102.3.3 Data()

```
virtual AAX_Result AAX_IDataBufferWrapper::Data (
    void const ** oBuffer ) const [pure virtual]
```

The buffer of data

Implemented in [AAX_VDataBufferWrapper](#).

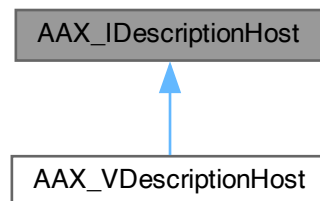
The documentation for this class was generated from the following file:

- [AAX_IDataBufferWrapper.h](#)

14.103 AAX_IDescriptionHost Class Reference

```
#include <AAX_IDescriptionHost.h>
```

Inheritance diagram for AAX_IDescriptionHost:



14.103.1 Description

Interface to host services provided during plug-in description

Public Member Functions

- virtual [~AAX_IDescriptionHost](#) ()
- virtual const [AAX_IFeatureInfo](#) * [AcquireFeatureProperties](#) (const [AAX_Feature_UID](#) &inFeatureID) const =0

14.103.2 Constructor & Destructor Documentation

14.103.2.1 ~AAX_IDescriptionHost()

```
virtual AAX_IDescriptionHost::~~AAX_IDescriptionHost ( ) [inline], [virtual]
```

14.103.3 Member Function Documentation

14.103.3.1 AcquireFeatureProperties()

```
virtual const AAX_IFeatureInfo * AAX_IDescriptionHost::AcquireFeatureProperties (
    const AAX_Feature_UID & inFeatureID ) const [pure virtual]
```

Get the client's feature object for a given feature ID

Similar to `QueryInterface()` but uses a feature identifier rather than a true IID

Ownership of the returned object is passed to the caller; the caller is responsible for destroying the object, e.g. by capturing the returned object in a smart pointer.

```
// AAX_IDescriptionHost* descHost
std::unique_ptr<const AAX_IFeatureInfo> featureInfoPtr(descHost->AcquireFeatureProperties(someFeatureUID);
```

Returns

An [AAX_IFeatureInfo](#) interface with access to the host's feature properties for this feature.

NULL if the desired feature was not found or if an error occurred

Note

May return an [AAX_IFeatureInfo](#) object with limited method support, which would return an error such as [AAX_ERROR_NULL_OBJECT](#) or [AAX_ERROR_UNIMPLEMENTED](#) to interface calls.

If no [AAX_IFeatureInfo](#) is provided then that may mean that the host is unaware of the feature, or it may mean that the host is aware of the feature but has not implemented the AAX feature support interface for this feature yet.

Parameters

in	<i>inFeatureID</i>	Identifier of the requested feature
----	--------------------	-------------------------------------

Implemented in [AAX_VDescriptionHost](#).

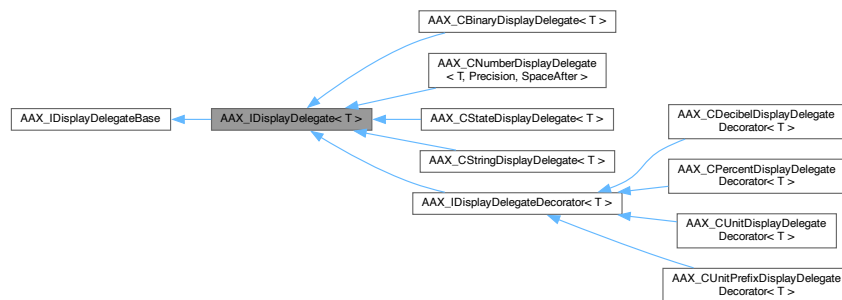
The documentation for this class was generated from the following file:

- [AAX_IDescriptionHost.h](#)

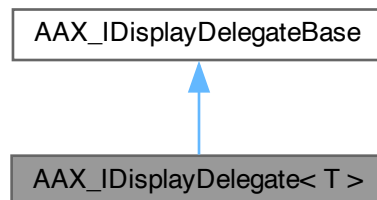
14.104 AAX_IDisplayDelegate< T > Class Template Reference

```
#include <AAX_IDisplayDelegate.h>
```

Inheritance diagram for `AAX_IDisplayDelegate< T >`:



Collaboration diagram for `AAX_IDisplayDelegate< T >`:



14.104.1 Description

```

template<typename T>
class AAX_IDisplayDelegate< T >

```

Display delegate interface template

Classes for parameter value string conversion.

Display delegates are used to convert real parameter values to and from their formatted string representations. All display delegates implement the [AAX_IDisplayDelegate](#) interface, which contains two conversion functions:

```

virtual bool ValueToString(T value, std::string& valueString) const = 0;
virtual bool StringToValue(const std::string& valueString, T& value) const = 0;

```

14.104.2 Display delegate decorators

The AAX SDK utilizes a decorator pattern in order to provide code re-use while accounting for a wide variety of possible parameter display formats. The SDK includes a number of sample display delegate decorator classes.

Each concrete display delegate decorator implements [AAX_IDisplayDelegateDecorator](#) and adheres to the decorator pattern. The decorator pattern allows multiple display behaviors to be composited or wrapped together at run time. For instance it is possible to implement a dBV (dB Volts) decorator, by wrapping an [AAX_CDecibelDisplayDelegateDecorator](#) with an [AAX_CUnitDisplayDelegateDecorator](#).

14.104.2.1 Display delegate decorator implementation

By implementing [AAX_IDisplayDelegateDecorator](#), each concrete display delegate decorator class implements the full [AAX_IDisplayDelegate](#) interface. In addition, it retains a pointer to the [AAX_IDisplayDelegateDecorator](#) that it wraps. When the decorator performs a conversion, it calls into its wrapped class so that the wrapped decorator may apply its own conversion formatting. By repeating this pattern in each decorator, all of the decorator subclasses call into their "wrapper" in turn, resulting in a final string to which all of the decorators' conversions have been applied in sequence.

Here is the relevant implementation from [AAX_IDisplayDelegateDecorator](#) :

```
template <typename T>
AAX_IDisplayDelegateDecorator<T>::AAX_IDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>&
    displayDelegate) :
    AAX_IDisplayDelegate<T>(),
    mWrappedDisplayDelegate(displayDelegate.Clone())
{
}

template <typename T>
bool AAX_IDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString) const
{
    return mWrappedDisplayDelegate->ValueToString(value, valueString);
}

template <typename T>
bool AAX_IDisplayDelegateDecorator<T>::StringToValue(const AAX_CString& valueString, T* value) const
{
    return mWrappedDisplayDelegate->StringToValue(valueString, value);
}
```

14.104.2.2 Decibel decorator example

Here is a concrete example of how a decibel decorator might be implemented

```
template <typename T>
bool AAX_CDecibelDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString) const
{
    if (value <= 0)
    {
        *valueString = AAX_CString("--- dB");
        return true;
    }

    value = 20*log10(value);
    bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
    *valueString += AAX_CString("dB");
    return succeeded;
}
```

Notice in this example that the [ValueToString\(\)](#) method is called in the parent class, [AAX_IDisplayDelegateDecorator](#). This results in a call into the wrapped class' implementation of [ValueToString\(\)](#), which converts the decorated value to a redecorated string, and so forth for additional decorators.

Public Member Functions

- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0
Constructs and returns a copy of the display delegate.
- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation.
- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0
Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.
- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0
Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.104.3 Member Function Documentation

14.104.3.1 Clone()

```
template<typename T >
virtual AAX\_IDisplayDelegate * AAX\_IDisplayDelegate< T >::Clone ( ) const [pure virtual]
```

Constructs and returns a copy of the display delegate.

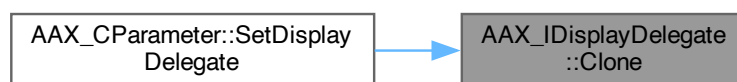
In general, this method's implementation can use a simple copy constructor:

```
template<typename T>
AAX_CSubclassDisplayDelegate<T>*    AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Implemented in [AAX_CBinaryDisplayDelegate< T >](#), [AAX_CDecibelDisplayDelegateDecorator< T >](#), [AAX_CNumberDisplayDelegateDecorator< T >](#), [AAX_CPercentDisplayDelegateDecorator< T >](#), [AAX_CStateDisplayDelegate< T >](#), [AAX_CStringDisplayDelegate< T >](#), [AAX_CUnitDisplayDelegateDecorator< T >](#), [AAX_CUnitPrefixDisplayDelegateDecorator< T >](#), and [AAX_IDisplayDelegateDecorator< T >](#).

Referenced by [AAX_CParameter< T >::SetDisplayDelegate\(\)](#).

Here is the caller graph for this function:



14.104.3.2 ValueToString() [1/2]

```
template<typename T >
virtual bool AAX\_IDisplayDelegate< T >::ValueToString (
    T value,
    AAX\_CString * valueString ) const [pure virtual]
```

Converts a real parameter value to a string representation.

Parameters

in	<i>value</i>	The real parameter value that will be converted
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CBinaryDisplayDelegate< T >](#), [AAX_CDecibelDisplayDelegateDecorator< T >](#), [AAX_CNumberDisplayDelegateDecorator< T >](#), [AAX_CPercentDisplayDelegateDecorator< T >](#), [AAX_CStateDisplayDelegate< T >](#), [AAX_CStringDisplayDelegate< T >](#), [AAX_CUnitDisplayDelegateDecorator< T >](#), [AAX_CUnitPrefixDisplayDelegateDecorator< T >](#), and [AAX_IDisplayDelegateDecorator< T >](#).

14.104.3.3 ValueToString() [2/2]

```
template<typename T >
virtual bool AAX_IDisplayDelegate< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [pure virtual]
```

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

Parameters

in	<i>value</i>	The real parameter value that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>valueString</i>	A string corresponding to value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CBinaryDisplayDelegate< T >](#), [AAX_CDecibelDisplayDelegateDecorator< T >](#), [AAX_CNumberDisplayDelegateDecorator< T >](#), [AAX_CPercentDisplayDelegateDecorator< T >](#), [AAX_CStateDisplayDelegate< T >](#), [AAX_CStringDisplayDelegate< T >](#), [AAX_CUnitDisplayDelegateDecorator< T >](#), [AAX_CUnitPrefixDisplayDelegateDecorator< T >](#), and [AAX_IDisplayDelegateDecorator< T >](#).

14.104.3.4 StringToValue()

```
template<typename T >
virtual bool AAX_IDisplayDelegate< T >::StringToValue (
```

```
const AAX_CString & valueString,
T * value ) const [pure virtual]
```

Converts a string to a real parameter value.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CBinaryDisplayDelegate< T >](#), [AAX_CDecibelDisplayDelegateDecorator< T >](#), [AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >](#), [AAX_CPercentDisplayDelegateDecorator< T >](#), [AAX_CStateDisplayDelegate< T >](#), [AAX_CStringDisplayDelegate< T >](#), [AAX_CUnitDisplayDelegateDecorator< T >](#), [AAX_CUnitPrefixDisplayDelegateDecorator< T >](#), and [AAX_IDisplayDelegateDecorator< T >](#).

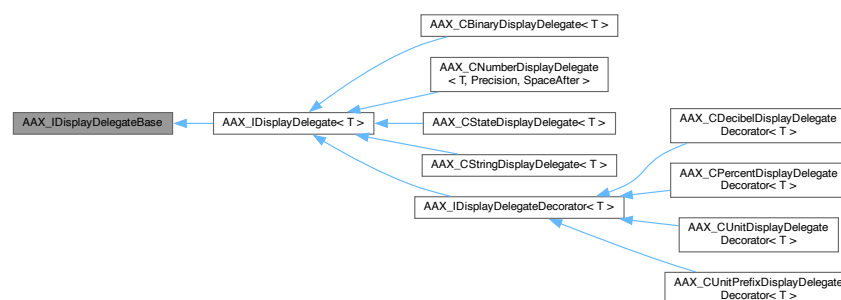
The documentation for this class was generated from the following file:

- [AAX_IDisplayDelegate.h](#)

14.105 AAX_IDisplayDelegateBase Class Reference

```
#include <AAX_IDisplayDelegate.h>
```

Inheritance diagram for AAX_IDisplayDelegateBase:



14.105.1 Description

Defines the display behavior for a parameter.

This interface represents a delegate class to be used in conjunction with [AAX_IParameter](#). [AAX_IParameter](#) delegates all conversion operations between strings and real parameter values to classes that meet this interface. You can think of [AAX_ITaperDelegate](#) subclasses as simple string serialization routines that enable a specific string conversions for an arbitrary parameter.

For more information about how parameter delegates operate, see the [AAX_ITaperDelegate](#) and [Parameter Manager](#) documentation.

Note

This class is *not* part of the AAX ABI and must not be passed between the plug-in and the host.

Public Member Functions

- virtual [~AAX_IDisplayDelegateBase](#) ()
Virtual destructor.

14.105.2 Constructor & Destructor Documentation**14.105.2.1 ~AAX_IDisplayDelegateBase()**

```
virtual AAX_IDisplayDelegateBase::~~AAX_IDisplayDelegateBase ( ) [inline], [virtual]
```

Virtual destructor.

Note

This destructor MUST be virtual to prevent memory leaks.

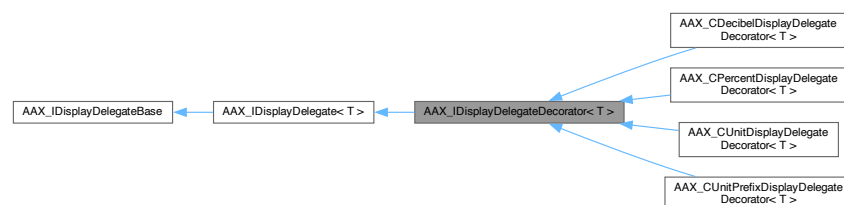
The documentation for this class was generated from the following file:

- [AAX_IDisplayDelegate.h](#)

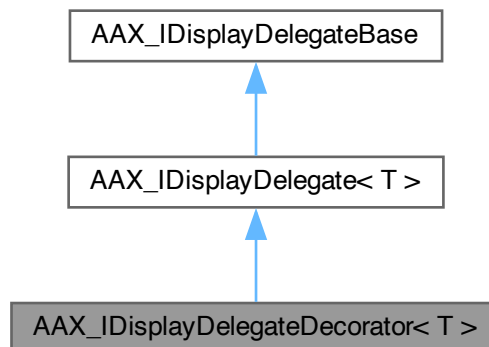
14.106 AAX_IDisplayDelegateDecorator< T > Class Template Reference

```
#include <AAX_IDisplayDelegateDecorator.h>
```

Inheritance diagram for AAX_IDisplayDelegateDecorator< T >:



Collaboration diagram for AAX_IDisplayDelegateDecorator< T >:



14.106.1 Description

```
template<typename T>
class AAX_IDisplayDelegateDecorator< T >
```

The base class for all concrete display delegate decorators.

The AAX parameter display strategy uses a decorator pattern for parameter value formatting. This approach allows developers to maximize code re-use across display delegates with many different kinds of varying formatting, all without creating interdependencies between the different display delegates themselves.

For more information, see [Display delegate decorators](#). For even more information, about the Decorator design pattern, please consult the GOF design patterns book.

Note

This class is *not* part of the AAX ABI and must not be passed between the plug-in and the host.

Public Member Functions

- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegate< T >](#) &displayDelegate)
Constructor.
- [AAX_IDisplayDelegateDecorator](#) (const [AAX_IDisplayDelegateDecorator](#) &other)
Copy constructor.
- [~AAX_IDisplayDelegateDecorator](#) () [AAX_OVERRIDE](#)
Virtual destructor.
- [AAX_IDisplayDelegateDecorator< T > * Clone](#) () const [AAX_OVERRIDE](#)
Constructs and returns a copy of the display delegate decorator.
- bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)
Converts a string to a real parameter value.
- bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const [AAX_OVERRIDE](#)

Converts a string to a real parameter value with a size constraint.

- bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const [AAX_OVERRIDE](#)

Converts a string to a real parameter value.

- virtual [AAX_IDisplayDelegate](#) * [Clone](#) () const =0

Constructs and returns a copy of the display delegate.

- virtual bool [ValueToString](#) (T value, [AAX_CString](#) *valueString) const =0

Converts a real parameter value to a string representation.

- virtual bool [ValueToString](#) (T value, int32_t maxNumChars, [AAX_CString](#) *valueString) const =0

Converts a real parameter value to a string representation using a size hint, useful for control surfaces and other character limited displays.

- virtual bool [StringToValue](#) (const [AAX_CString](#) &valueString, T *value) const =0

Converts a string to a real parameter value.

Public Member Functions inherited from [AAX_IDisplayDelegateBase](#)

- virtual [~AAX_IDisplayDelegateBase](#) ()

Virtual destructor.

14.106.2 Constructor & Destructor Documentation

14.106.2.1 [AAX_IDisplayDelegateDecorator\(\)](#) [1/2]

```
template<typename T >
AAX_IDisplayDelegateDecorator< T >::AAX_IDisplayDelegateDecorator (
    const AAX_IDisplayDelegate< T > & displayDelegate )
```

Constructor.

This class implements the decorator pattern, which is a sort of wrapper. The object that is being wrapped is passed into this constructor. This object is passed by reference because it must be copied to prevent any potential memory ambiguities.

This constructor sets the local mWrappedDisplayDelegate member to a clone of the provided [AAX_IDisplayDelegate](#).

Parameters

<code>in</code>	<code>displayDelegate</code>	The decorated display delegate.
-----------------	------------------------------	---------------------------------

14.106.2.2 [AAX_IDisplayDelegateDecorator\(\)](#) [2/2]

```
template<typename T >
AAX_IDisplayDelegateDecorator< T >::AAX_IDisplayDelegateDecorator (
    const AAX_IDisplayDelegateDecorator< T > & other )
```

Copy constructor.

This class implements the decorator pattern, which is a sort of wrapper. The object that is being wrapped is passed into this constructor. This object is passed by reference because it must be copied to prevent any potential memory ambiguities.

This constructor sets the local `mWrappedDisplayDelegate` member to a clone of the provided [AAX_IDisplayDelegateDecorator](#), allowing multiply-decorated display delegates.

Parameters

<i>in</i>	<i>other</i>	The display delegate decorator that will be set as the wrapped delegate of this object
-----------	--------------	--

14.106.2.3 ~AAX_IDisplayDelegateDecorator()

```
template<typename T >
AAX_IDisplayDelegateDecorator< T >::~~AAX_IDisplayDelegateDecorator
```

Virtual destructor.

Note

This destructor must be overridden here in order to delete the wrapped display delegate object upon decorator destruction.

14.106.3 Member Function Documentation

14.106.3.1 Clone()

```
template<typename T >
AAX_IDisplayDelegateDecorator< T > * AAX_IDisplayDelegateDecorator< T >::Clone() [virtual]
```

Constructs and returns a copy of the display delegate decorator.

In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassDisplayDelegate<T>* AAX_CSubclassDisplayDelegate<T>::Clone() const
{
    return new AAX_CSubclassDisplayDelegate(*this);
}
```

Note

This is an idiomatic method in the decorator pattern, so watch for potential problems if this method is ever changed or removed.

Implements [AAX_IDisplayDelegate< T >](#).

14.106.3.2 ValueToString() [1/2]

```
template<typename T >
bool AAX_IDisplayDelegateDecorator< T >::ValueToString (
    T value,
    AAX_CString * valueString ) const [virtual]
```

Converts a string to a real parameter value.

Override of the [AAX_IDisplayDelegate](#) implementation to call into the wrapped object. Display delegate decorators should call into this implementation to pass [ValueToString\(\)](#) calls on to the wrapped object after applying their own value-to-string decoration.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

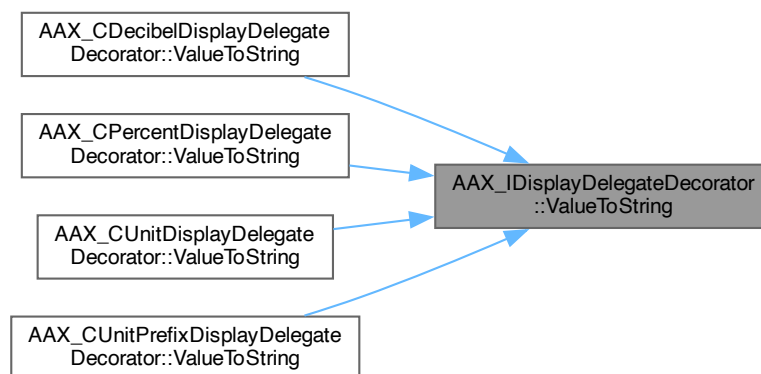
Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

Referenced by [AAX_CDecibelDisplayDelegateDecorator< T >::ValueToString\(\)](#), [AAX_CPercentDisplayDelegateDecorator< T >::ValueToString\(\)](#), [AAX_CUnitDisplayDelegateDecorator< T >::ValueToString\(\)](#), and [AAX_CUnitPrefixDisplayDelegateDecorator< T >::ValueToString\(\)](#).

Here is the caller graph for this function:



14.106.3.3 ValueToString() [2/2]

```
template<typename T >
bool AAX_IDisplayDelegateDecorator< T >::ValueToString (
    T value,
    int32_t maxNumChars,
    AAX_CString * valueString ) const [virtual]
```

Converts a string to a real parameter value with a size constraint.

Override of the [AAX_IDisplayDelegate](#) implementation to call into the wrapped object. Display delegate decorators should call into this implementation to pass [ValueToString\(\)](#) calls on to the wrapped object after applying their own value-to-string decoration.

Parameters

in	<i>valueString</i>	The string that will be converted
in	<i>maxNumChars</i>	Size hint for the desired maximum number of characters in the string (not including null termination)
out	<i>value</i>	The real parameter value corresponding to valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implements [AAX_IDisplayDelegate< T >](#).

14.106.3.4 StringToValue()

```
template<typename T >
bool AAX_IDisplayDelegateDecorator< T >::StringToValue (
    const AAX_CString & valueString,
    T * value ) const [virtual]
```

Converts a string to a real parameter value.

Override of the DisplayDecorator implementation to call into the wrapped object. Display delegate decorators should call into this implementation to pass [StringToValue\(\)](#) calls on to the wrapped object after applying their own string-to-value decoding.

Parameters

in	<i>valueString</i>	The string that will be converted
out	<i>value</i>	The real parameter value corresponding to valueString

Return values

<i>true</i>	The string conversion was successful
-------------	--------------------------------------

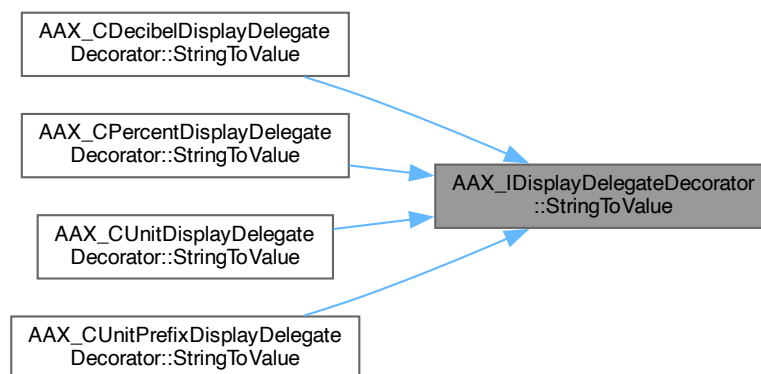
Return values

<i>false</i>	The string conversion was unsuccessful
--------------	--

Implements [AAX_IDisplayDelegate< T >](#).

Referenced by [AAX_CDecibelDisplayDelegateDecorator< T >::StringToValue\(\)](#), [AAX_CPercentDisplayDelegateDecorator< T >::StringToValue\(\)](#), [AAX_CUnitDisplayDelegateDecorator< T >::StringToValue\(\)](#), and [AAX_CUnitPrefixDisplayDelegateDecorator< T >::StringToValue\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [AAX_IDisplayDelegateDecorator.h](#)

14.107 AAX_IDma Class Reference

```
#include <AAX_IDma.h>
```

14.107.1 Description

Cross-platform interface for access to the host's direct memory access (DMA) facilities.

:Implemented by the AAX Host

This interface is provided via a DMA port in the plug-in's algorithm context.

See also

[AAX_IComponentDescriptor::AddDmaInstance\(\)](#)
[Direct Memory Access](#)

Public Types

- enum `EState` {
`eState_Error` = -1 ,
`eState_Init` = 0 ,
`eState_Running` = 1 ,
`eState_Complete` = 2 ,
`eState_Pending` = 3 }
- enum `EMode` {
`eMode_Error` = -1 ,
`eMode_Burst` = 6 ,
`eMode_Gather` = 10 ,
`eMode_Scatter` = 11 }

DMA mode IDs.

Public Member Functions

- virtual `~AAX_IDma` ()

Basic DMA operation

- virtual `AAX_Result AAX_DMA_API PostRequest` ()=0
Posts the transfer request to the DMA server.
- virtual `int32_t AAX_DMA_API IsTransferComplete` ()=0
Query whether a transfer has completed.
- virtual `AAX_Result AAX_DMA_API SetDmaState` (`EState` iState)=0
Sets the DMA State.
- virtual `EState AAX_DMA_API GetDmaState` () const =0
Inquire to find the state of the DMA instance.
- virtual `EMode AAX_DMA_API GetDmaMode` () const =0
Inquire to find the mode of the DMA instance.

Methods for Burst operation

Use these methods in conjunction with `AAX_IDma::eMode_Burst`

- virtual `AAX_Result AAX_DMA_API SetSrc` (`int8_t *iSrc`)=0
Sets the address of the source buffer.
- virtual `int8_t *AAX_DMA_API GetSrc` ()=0
Gets the address of the source buffer.
- virtual `AAX_Result AAX_DMA_API SetDst` (`int8_t *iDst`)=0
Sets the address of the destination buffer.
- virtual `int8_t *AAX_DMA_API GetDst` ()=0
Gets the address of the destination buffer.
- virtual `AAX_Result AAX_DMA_API SetBurstLength` (`int32_t iBurstLengthBytes`)=0
Sets the length of each burst.
- virtual `int32_t AAX_DMA_API GetBurstLength` ()=0
Gets the length of each burst.
- virtual `AAX_Result AAX_DMA_API SetNumBursts` (`int32_t iNumBursts`)=0
Sets the number of bursts to perform before giving up priority to other DMA transfers.
- virtual `int32_t AAX_DMA_API GetNumBursts` ()=0
Gets the number of bursts to perform before giving up priority to other DMA transfers.
- virtual `AAX_Result AAX_DMA_API SetTransferSize` (`int32_t iTransferSizeBytes`)=0
Sets the size of the whole transfer.
- virtual `int32_t AAX_DMA_API GetTransferSize` ()=0
Gets the size of the whole transfer, in Bytes.

Methods for Scatter and Gather operation

Use these methods in conjunction with [AAX_IDma::eMode_Scatter](#) and [AAX_IDma::eMode_Gather](#)

- virtual [AAX_Result AAX_DMA_API SetFifoBuffer](#) (int8_t *iFifoBase)=0
Sets the address of the FIFO buffer for the DMA transfer (usually the external memory block)
- virtual int8_t *[AAX_DMA_API GetFifoBuffer](#) ()=0
Gets the address of the FIFO buffer for the DMA transfer.
- virtual [AAX_Result AAX_DMA_API SetLinearBuffer](#) (int8_t *iLinearBase)=0
Sets the address of the linear buffer for the DMA transfer (usually the internal memory block)
- virtual int8_t *[AAX_DMA_API GetLinearBuffer](#) ()=0
Gets the address of the linear buffer for the DMA transfer.
- virtual [AAX_Result AAX_DMA_API SetOffsetTable](#) (const int32_t *iOffsetTable)=0
Sets the offset table for the DMA transfer.
- virtual const int32_t *[AAX_DMA_API GetOffsetTable](#) ()=0
Gets the offset table for the DMA transfer.
- virtual [AAX_Result AAX_DMA_API SetNumOffsets](#) (int32_t iNumOffsets)=0
Sets the number of offsets in the offset table.
- virtual int32_t [AAX_DMA_API GetNumOffsets](#) ()=0
Gets the number of offsets in the offset table.
- virtual [AAX_Result AAX_DMA_API SetBaseOffset](#) (int32_t iBaseOffsetBytes)=0
Sets the relative base offset into the FIFO where transfers will begin.
- virtual int32_t [AAX_DMA_API GetBaseOffset](#) ()=0
Gets the relative base offset into the FIFO where transfers will begin.
- virtual [AAX_Result AAX_DMA_API SetFifoSize](#) (int32_t iSizeBytes)=0
Sets the size of the FIFO buffer, in bytes.
- virtual int32_t [AAX_DMA_API GetFifoSize](#) ()=0
Gets the size of the FIFO buffer, in bytes.

14.107.2 Member Enumeration Documentation

14.107.2.1 EState

```
enum AAX_IDma::EState
```

Enumerator

eState_Error	
eState_Init	
eState_Running	
eState_Complete	
eState_Pending	

14.107.2.2 EMode

```
enum AAX_IDma::EMode
```

DMA mode IDs.

These IDs are used to bind DMA context fields to a particular DMA mode when describing the fields with [AAX_IComponentDescriptor::AddDmaInstance\(\)](#)

Enumerator

eMode_Error	
eMode_Burst	Burst mode (uncommon)
eMode_Gather	Gather mode.
eMode_Scatter	Scatter mode.

14.107.3 Constructor & Destructor Documentation

14.107.3.1 ~AAX_IDma()

```
virtual AAX_IDma::~~AAX_IDma ( ) [inline], [virtual]
```

14.107.4 Member Function Documentation

14.107.4.1 PostRequest()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::PostRequest ( ) [pure virtual]
```

Posts the transfer request to the DMA server.

Note

Whichever mode this method is called on first will be the first mode to start transferring. Most plug-ins should therefore call this method for their Scatter DMA fields before their Gather DMA fields so that the scattered data is available as quickly as possible for future gathers.

Returns

AAX_SUCCESS on success

14.107.4.2 IsTransferComplete()

```
virtual int32_t AAX_DMA_API AAX_IDma::IsTransferComplete ( ) [pure virtual]
```

Query whether a transfer has completed.

A return value of false indicates an error, and that the DMA missed its cycle count deadline

Note

This function should not be used for polling within a Process loop! Instead, it can be used as a test for DMA failure. This test is usually performed via a Debug-only assert.

Todo Clarify return value meaning – ambiguity in documentation

Returns

true if all pending transfers are complete

false if pending transfers are not complete

14.107.4.3 SetDmaState()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetDmaState (
    EState iState ) [pure virtual]
```

Sets the DMA State.

Note

This method is part of the host interface and should not be used by plug-ins

Returns

AAX_SUCCESS on success

14.107.4.4 GetDmaState()

```
virtual EState AAX_DMA_API AAX_IDma::GetDmaState ( ) const [pure virtual]
```

Inquire to find the state of the DMA instance.

14.107.4.5 GetDmaMode()

```
virtual EMode AAX_DMA_API AAX_IDma::GetDmaMode ( ) const [pure virtual]
```

Inquire to find the mode of the DMA instance.

This value does not change, so there is no setter.

14.107.4.6 SetSrc()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetSrc (
    int8_t * iSrc ) [pure virtual]
```

Sets the address of the source buffer.

Parameters

<i>in</i>	<i>iSrc</i>	Address of the location in the source buffer where the read transfer should begin
-----------	-------------	---

Returns

AAX_SUCCESS on success

14.107.4.7 GetSrc()

```
virtual int8_t *AAX_DMA_API AAX_IDma::GetSrc ( ) [pure virtual]
```

Gets the address of the source buffer.

14.107.4.8 SetDst()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetDst (
    int8_t * iDst ) [pure virtual]
```

Sets the address of the destination buffer.

Parameters

<i>in</i>	<i>iDst</i>	Address of the location in the destination buffer where the write transfer should begin
-----------	-------------	---

Returns

AAX_SUCCESS on success

14.107.4.9 GetDst()

```
virtual int8_t *AAX_DMA_API AAX_IDma::GetDst ( ) [pure virtual]
```

Gets the address of the destination buffer.

14.107.4.10 SetBurstLength()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetBurstLength (
    int32_t iBurstLengthBytes ) [pure virtual]
```

Sets the length of each burst.

Note

Burst length must be between 1 and 64 Bytes, inclusive

64-Byte transfers are recommended for the fastest overall transfer speed

Returns

AAX_SUCCESS on success

14.107.4.11 GetBurstLength()

```
virtual int32_t AAX_DMA_API AAX_IDma::GetBurstLength ( ) [pure virtual]
```

Gets the length of each burst.

14.107.4.12 SetNumBursts()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetNumBursts (
    int32_t iNumBursts ) [pure virtual]
```

Sets the number of bursts to perform before giving up priority to other DMA transfers.

Valid values are 1, 2, 4, or 16.

The full transmission may be broken up into several series of bursts, and thus the total size of the data being transferred is not bounded by the number of bursts times the burst length.

Parameters

in	<i>iNumBursts</i>	The number of bursts
----	-------------------	----------------------

Returns

AAX_SUCCESS on success

14.107.4.13 GetNumBursts()

```
virtual int32_t AAX_DMA_API AAX_IDma::GetNumBursts ( ) [pure virtual]
```

Gets the number of bursts to perform before giving up priority to other DMA transfers.

14.107.4.14 SetTransferSize()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetTransferSize (
    int32_t iTransferSizeBytes ) [pure virtual]
```

Sets the size of the whole transfer.

Parameters

in	<i>iTransferSizeBytes</i>	The transfer size, in Bytes
----	---------------------------	-----------------------------

Returns

AAX_SUCCESS on success

14.107.4.15 GetTransferSize()

```
virtual int32_t AAX_DMA_API AAX_IDma::GetTransferSize ( ) [pure virtual]
```

Gets the size of the whole transfer, in Bytes.

14.107.4.16 SetFifoBuffer()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetFifoBuffer (
    int8_t * iFifoBase ) [pure virtual]
```

Sets the address of the FIFO buffer for the DMA transfer (usually the external memory block)

Returns

AAX_SUCCESS on success

14.107.4.17 GetFifoBuffer()

```
virtual int8_t *AAX_DMA_API AAX_IDma::GetFifoBuffer ( ) [pure virtual]
```

Gets the address of the FIFO buffer for the DMA transfer.

14.107.4.18 SetLinearBuffer()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetLinearBuffer (
    int8_t * iLinearBase ) [pure virtual]
```

Sets the address of the linear buffer for the DMA transfer (usually the internal memory block)

Returns

AAX_SUCCESS on success

14.107.4.19 GetLinearBuffer()

```
virtual int8_t *AAX_DMA_API AAX_IDma::GetLinearBuffer ( ) [pure virtual]
```

Gets the address of the linear buffer for the DMA transfer.

14.107.4.20 SetOffsetTable()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetOffsetTable (
    const int32_t * iOffsetTable ) [pure virtual]
```

Sets the offset table for the DMA transfer.

The offset table provides a list of Byte-aligned memory offsets into the FIFO buffer. The transfer will be broken into a series of individual bursts, each beginning at the specified offset locations within the FIFO buffer. The size of each burst is set by [SetBurstLength\(\)](#).

See also

[AAX_IDma::SetNumOffsets\(\)](#)

[AAX_IDma::SetBaseOffset\(\)](#)

Returns

AAX_SUCCESS on success

14.107.4.21 GetOffsetTable()

```
virtual const int32_t *AAX_DMA_API AAX_IDma::GetOffsetTable ( ) [pure virtual]
```

Gets the offset table for the DMA transfer.

14.107.4.22 SetNumOffsets()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetNumOffsets (
    int32_t iNumOffsets ) [pure virtual]
```

Sets the number of offsets in the offset table.

See also

[AAX_IDma::SetOffsetTable\(\)](#)

Returns

AAX_SUCCESS on success

14.107.4.23 GetNumOffsets()

```
virtual int32_t AAX_DMA_API AAX_IDma::GetNumOffsets ( ) [pure virtual]
```

Gets the number of offsets in the offset table.

14.107.4.24 SetBaseOffset()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetBaseOffset (
    int32_t iBaseOffsetBytes ) [pure virtual]
```

Sets the relative base offset into the FIFO where transfers will begin.

The base offset will be added to each value in the offset table in order to determine the starting offset within the FIFO buffer for each burst.

See also

[AAX_IDma::SetOffsetTable\(\)](#)

Returns

AAX_SUCCESS on success

14.107.4.25 GetBaseOffset()

```
virtual int32_t AAX_DMA_API AAX_IDma::GetBaseOffset ( ) [pure virtual]
```

Gets the relative base offset into the FIFO where transfers will begin.

14.107.4.26 SetFifoSize()

```
virtual AAX_Result AAX_DMA_API AAX_IDma::SetFifoSize (
    int32_t iSizeBytes ) [pure virtual]
```

Sets the size of the FIFO buffer, in bytes.

Note

The FIFO buffer must be padded with at least enough memory to accommodate one burst, as defined by [SetBurstLength\(\)](#).

Returns

AAX_SUCCESS on success

14.107.4.27 GetFifoSize()

```
virtual int32_t AAX_DMA_API AAX_IDma::GetFifoSize ( ) [pure virtual]
```

Gets the size of the FIFO buffer, in bytes.

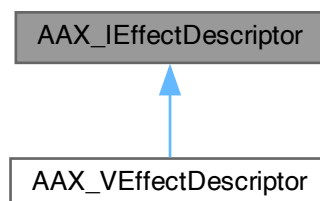
The documentation for this class was generated from the following file:

- [AAX_IDma.h](#)

14.108 AAX_IEffectDescriptor Class Reference

```
#include <AAX_IEffectDescriptor.h>
```

Inheritance diagram for AAX_IEffectDescriptor:



14.108.1 Description

Description interface for an effect's (plug-in type's) components.

:Implemented by the AAX Host

Each Effect represents a different "type" of plug-in. The host will present different Effects to the user as separate products, even if they are derived from the same [AAX_ICollection](#) description.

See also

[AAX_ICollection::AddEffect\(\)](#)

Public Member Functions

- virtual [~AAX_IEffectDescriptor](#) ()
- virtual [AAX_IComponentDescriptor](#) * [NewComponentDescriptor](#) ()=0
Create an instance of a component descriptor.
- virtual [AAX_Result](#) [AddComponent](#) ([AAX_IComponentDescriptor](#) *inComponentDescriptor)=0
Add a component to an instance of a component descriptor.
- virtual [AAX_Result](#) [AddName](#) (const char *inPlugInName)=0
Add a name to the Effect.
- virtual [AAX_Result](#) [AddCategory](#) (uint32_t inCategory)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result](#) [AddCategoryBypassParameter](#) (uint32_t inCategory, [AAX_CParamID](#) inParamID)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result](#) [AddProcPtr](#) (void *inProcPtr, [AAX_CProcPtrID](#) inProcID)=0
Add a process pointer.
- virtual [AAX_IPropertyMap](#) * [NewPropertyMap](#) ()=0
Create a new property map.
- virtual [AAX_Result](#) [SetProperties](#) ([AAX_IPropertyMap](#) *inProperties)=0
Set the properties of a new property map.
- virtual [AAX_Result](#) [AddResourceInfo](#) ([AAX_EResourceType](#) inResourceType, const char *inInfo)=0
Set resource file info.
- virtual [AAX_Result](#) [AddMeterDescription](#) ([AAX_CTypeID](#) inMeterID, const char *inMeterName, [AAX_IPropertyMap](#) *inProperties)=0
Add name and property map to meter with given ID.
- virtual [AAX_Result](#) [AddControlMIDINode](#) ([AAX_CTypeID](#) inNodeID, [AAX_EMIDINodeType](#) inNodeType, const char inNodeName[], uint32_t inChannelMask)=0
Add a control MIDI node to the plug-in data model.

14.108.2 Constructor & Destructor Documentation

14.108.2.1 ~AAX_IEffectDescriptor()

```
virtual AAX_IEffectDescriptor::~AAX_IEffectDescriptor ( ) [inline], [virtual]
```

14.108.3 Member Function Documentation

14.108.3.1 NewComponentDescriptor()

```
virtual AAX\_IComponentDescriptor * AAX_IEffectDescriptor::NewComponentDescriptor ( ) [pure virtual]
```

Create an instance of a component descriptor.

Implemented in [AAX_VEffectDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.108.3.2 AddComponent()

```
virtual AAX\_Result AAX_IEffectDescriptor::AddComponent (
    AAX\_IComponentDescriptor * inComponentDescriptor ) [pure virtual]
```

Add a component to an instance of a component descriptor.

Unlike with [AAX_ICollection::AddEffect\(\)](#), the [AAX_IEffectDescriptor](#) does not take ownership of the [AAX_IComponentDescriptor](#) that is passed to it in this method. The host copies out the contents of this descriptor, and thus the plug-in may re-use the same descriptor object when creating additional similar components.

Parameters

in	<i>inComponentDescriptor</i>	
----	------------------------------	--

Implemented in [AAX_VEffectDescriptor](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:



14.108.3.3 AddName()

```
virtual AAX_Result AAX_IEffectDescriptor::AddName (
    const char * inPlugInName ) [pure virtual]
```

Add a name to the Effect.

May be called multiple times to add abbreviated Effect names.

Note

Every Effect must include at least one name variant with 31 or fewer characters, plus a null terminating character

Parameters

in	<i>inPlugInName</i>	The name assigned to the plug-in.
----	---------------------	-----------------------------------

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.4 AddCategory()

```
virtual AAX_Result AAX_IEffectDescriptor::AddCategory (
    uint32_t inCategory ) [pure virtual]
```

Add a category to your plug-in. See [AAX_EPlugInCategory](#).

Parameters

in	<i>inCategory</i>	One of the categories for the plug-in.
----	-------------------	--

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.5 AddCategoryBypassParameter()

```
virtual AAX_Result AAX_IEffectDescriptor::AddCategoryBypassParameter (
    uint32_t inCategory,
    AAX_CParamID inParamID ) [pure virtual]
```

Add a category to your plug-in. See [AAX_EPlugInCategory](#).

Parameters

in	<i>inCategory</i>	One of the categories for the plug-in.
in	<i>inParamID</i>	The parameter ID of the parameter used to bypass the category section of the plug-in.

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.6 AddProcPtr()

```
virtual AAX_Result AAX_IEffectDescriptor::AddProcPtr (
    void * inProcPtr,
    AAX_CProcPtrID inProcID ) [pure virtual]
```

Add a process pointer.

Parameters

in	<i>inProcPtr</i>	A process pointer.
in	<i>inProcID</i>	A process ID.

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.7 NewPropertyMap()

```
virtual AAX_IPropertyMap * AAX_IEffectDescriptor::NewPropertyMap ( ) [pure virtual]
```

Create a new property map.

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.8 SetProperties()

```
virtual AAX_Result AAX_IEffectDescriptor::SetProperties (
    AAX_IPropertyMap * inProperties ) [pure virtual]
```

Set the properties of a new property map.

Parameters

in	<i>inProperties</i>	Description
----	---------------------	-------------

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.9 AddResourceInfo()

```
virtual AAX_Result AAX_IEffectDescriptor::AddResourceInfo (
    AAX_EResourceType inResourceType,
    const char * inInfo ) [pure virtual]
```

Set resource file info.

Parameters

in	<i>inResourceType</i>	See AAX_EResourceType.
in	<i>inInfo</i>	Definition varies on the resource type.

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.10 AddMeterDescription()

```
virtual AAX_Result AAX_IEffectDescriptor::AddMeterDescription (
    AAX_CTypeID inMeterID,
    const char * inMeterName,
    AAX_IPropertyMap * inProperties ) [pure virtual]
```

Add name and property map to meter with given ID.

Parameters

in	<i>inMeterID</i>	The ID of the meter being described.
in	<i>inMeterName</i>	The name of the meter.
in	<i>inProperties</i>	The property map containing meter related data such as meter type, orientation, etc.

Implemented in [AAX_VEffectDescriptor](#).

14.108.3.11 AddControlMIDINode()

```
virtual AAX_Result AAX_IEffectDescriptor::AddControlMIDINode (
    AAX_CTypeID inNodeID,
```

```
AAX_EMIDINodeType inNodeType,
const char inNodeName[],
uint32_t inChannelMask ) [pure virtual]
```

Add a control MIDI node to the plug-in data model.

- This MIDI node may receive note data as well as control data.
- To send MIDI data to the plug-in's algorithm, use [AAX_IComponentDescriptor::AddMIDINode\(\)](#).

See also

[AAX_IACFEffectParameters_V2::UpdateControlMIDINodes\(\)](#)

Parameters

in	<i>inNodeID</i>	The ID for the new control MIDI node.
in	<i>inNodeType</i>	The type of the node.
in	<i>inNodeName</i>	The name of the node.
in	<i>inChannelMask</i>	The bit mask for required nodes channels (up to 16) or required global events for global node.

Implemented in [AAX_VEffectDescriptor](#).

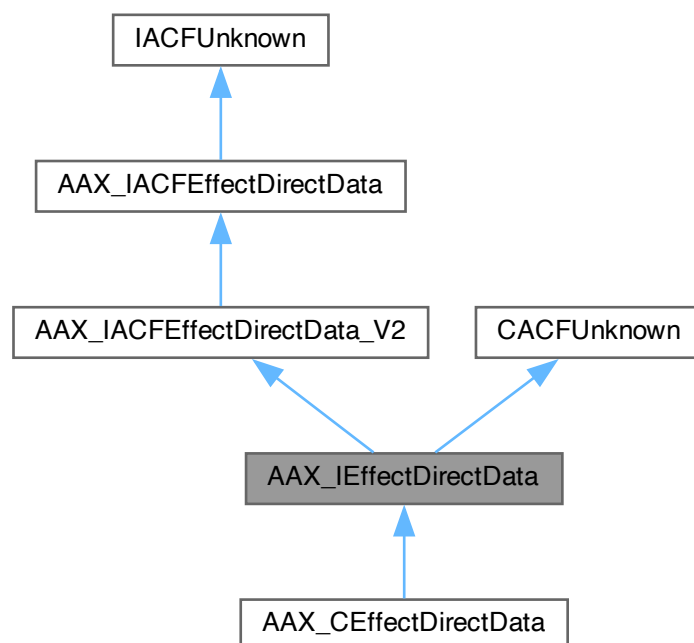
The documentation for this class was generated from the following file:

- [AAX_IEffectDescriptor.h](#)

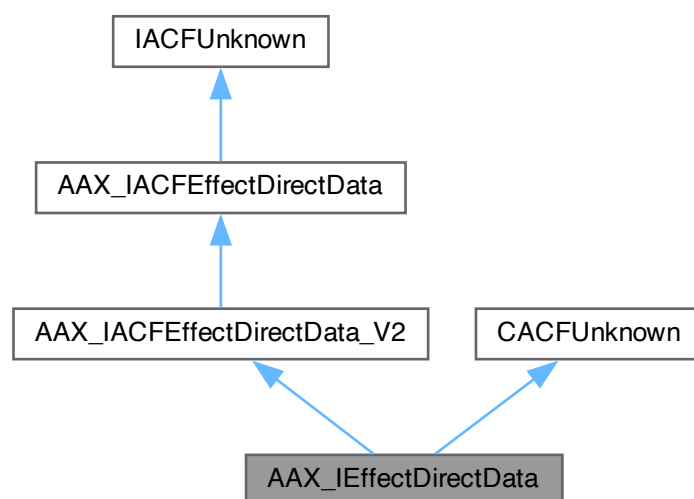
14.109 AAX_IEffectDirectData Class Reference

```
#include <AAX_IEffectDirectData.h>
```

Inheritance diagram for AAX_IEffectDirectData:



Collaboration diagram for AAX_IEffectDirectData:



14.109.1 Description

The interface for a AAX Plug-in's direct data interface.

:Implemented by the Plug-In

This is the interface for an instance of a plug-in's direct data interface that gets exposed to the host application. A plug-in needs to inherit from this interface and override all of the virtual functions to support direct data access functionality.

Direct data access allows a plug-in to directly manipulate the data in its algorithm's private data blocks. The callback methods in this interface provide a safe context from which this kind of access may be attempted.

Note

This class always inherits from the latest version of the interface and thus requires any subclass to implement all the methods in the latest version of the interface.

See [AAX_IACFEfffectDirectData](#) for further information.

Public Member Functions

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid
- [AAX_DELETE](#) ([AAX_IEfffectDirectData](#) &operator=(const [AAX_IEfffectDirectData](#) &))

Public Member Functions inherited from [AAX_IACFEfffectDirectData_V2](#)

- virtual [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.

Public Member Functions inherited from [AAX_IACFEfffectDirectData](#)

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Main initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Main uninitialization.
- virtual [AAX_Result TimerWakeup](#) ([IACFUnknown](#) *iDataAccessInterface)=0
Periodic wakeup callback for idle-time operations.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfiID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Public Attributes

- void **ppvObjOut [override](#)

14.109.2 Member Function Documentation

14.109.2.1 ACF_DECLARE_STANDARD_UNKNOWN()

```
AAX_IEffectDirectData::ACF_DECLARE_STANDARD_UNKNOWN ( ) const &
```

14.109.2.2 AAX_DELETE()

```
AAX_IEffectDirectData::AAX_DELETE (
    AAX_IEffectDirectData & operator = (const AAX_IEffectDirectData &) )
```

14.109.3 Member Data Documentation

14.109.3.1 override

```
void** ppvObjOut AAX_IEffectDirectData::override
```

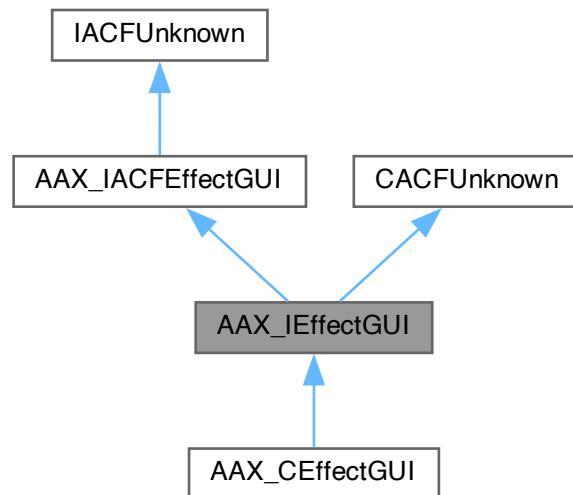
The documentation for this class was generated from the following file:

- [AAX_IEffectDirectData.h](#)

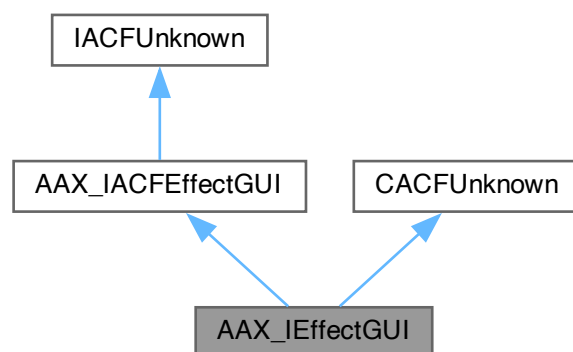
14.110 AAX_IEffectGUI Class Reference

```
#include <AAX_IEffectGUI.h>
```

Inheritance diagram for AAX_IEffectGUI:



Collaboration diagram for AAX_IEffectGUI:



14.110.1 Description

The interface for a AAX Plug-in's user interface.

:Implemented by the Plug-In

This is the interface for an instance of a plug-in's GUI that gets exposed to the host application. You need to inherit from this interface and override all of the virtual functions to create a plug-in GUI.

To create the GUI for an AAX plug-in it is required that you inherit from this interface and override all of the virtual functions from [AAX_IACFEffectGUI](#). In nearly all cases you will be able to take advantage of the implementations in the AAX library's [AAX_CEffectGUI](#) class and only override the few specific methods that you want to explicitly customize.

Note

This class always inherits from the latest version of the interface and thus requires any subclass to implement all the methods in the latest version of the interface.

Public Member Functions

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acflID](#) &riid
- [AAX_DELETE](#) (AAX_IEffectGUI &operator=(const [AAX_IEffectGUI](#) &))

Public Member Functions inherited from [AAX_IACFEffectGUI](#)

- virtual [AAX_Result Initialize](#) (IACFUnknown *iController)=0
Main GUI initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Main GUI uninitialization.
- virtual [AAX_Result NotificationReceived](#) (AAX_CTypeID inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.
- virtual [AAX_Result SetViewContainer](#) (IACFUnknown *iViewContainer)=0
Provides a handle to the main plug-in window.
- virtual [AAX_Result GetViewSize](#) (AAX_Point *oViewSize) const =0
Retrieves the size of the plug-in window.
- virtual [AAX_Result Draw](#) (AAX_Rect *iDrawRect)=0
DEPRECATED, Not called from host any longer. Your chosen graphics framework should be directly handling draw events from the OS.
- virtual [AAX_Result TimerWakeup](#) ()=0
Periodic wakeup callback for idle-time operations.
- virtual [AAX_Result ParameterUpdated](#) (AAX_CParamID inParamID)=0
Notifies the GUI that a parameter value has changed.
- virtual [AAX_Result GetCustomLabel](#) (AAX_EPlugInStrings iSelector, AAX_IString *oString) const =0
Called by host application to retrieve a custom plug-in string.
- virtual [AAX_Result SetControlHighlightInfo](#) (AAX_CParamID iParameterID, AAX_CBoolean iIsHighlighted, AAX_EHighlightColor iColor)=0
Called by host application. Indicates that a control widget should be updated with a highlight color.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Public Attributes

- void **ppvObjOut [override](#)

14.110.2 Member Function Documentation

14.110.2.1 ACF_DECLARE_STANDARD_UNKNOWN()

```
AAX_IEffectGUI::ACF_DECLARE_STANDARD_UNKNOWN ( ) const &
```

14.110.2.2 AAX_DELETE()

```
AAX_IEffectGUI::AAX_DELETE (
    AAX\_IEffectGUI & operator = (const AAX\_IEffectGUI &) )
```

14.110.3 Member Data Documentation

14.110.3.1 override

```
void** ppvObjOut AAX_IEffectGUI::override
```

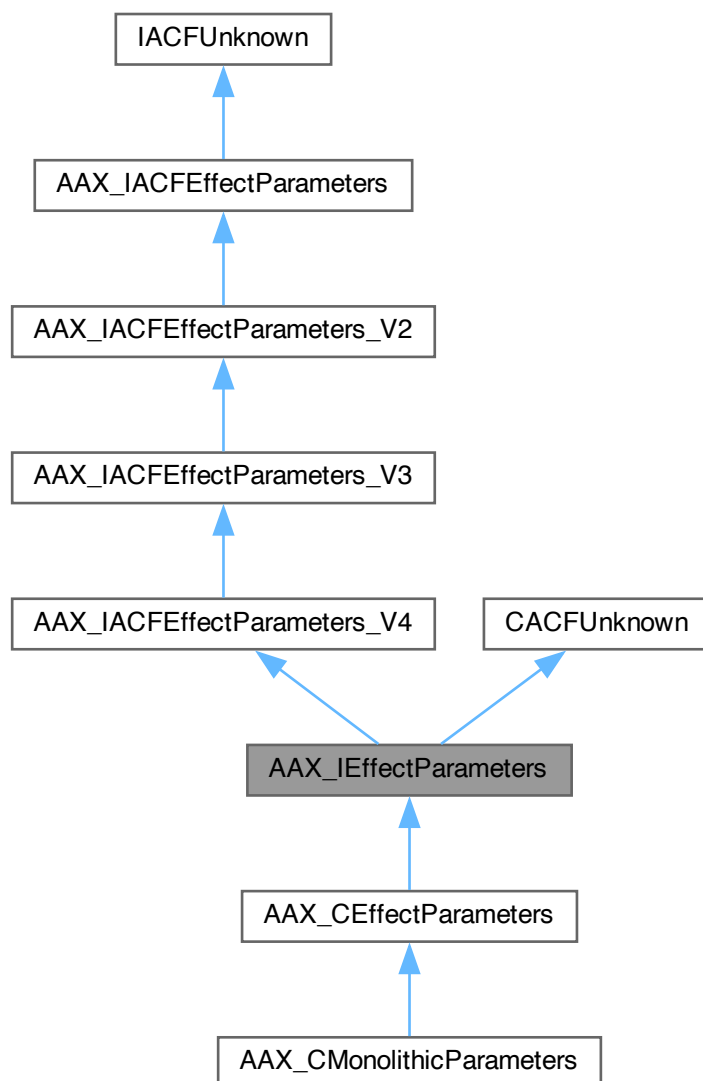
The documentation for this class was generated from the following file:

- [AAX_IEffectGUI.h](#)

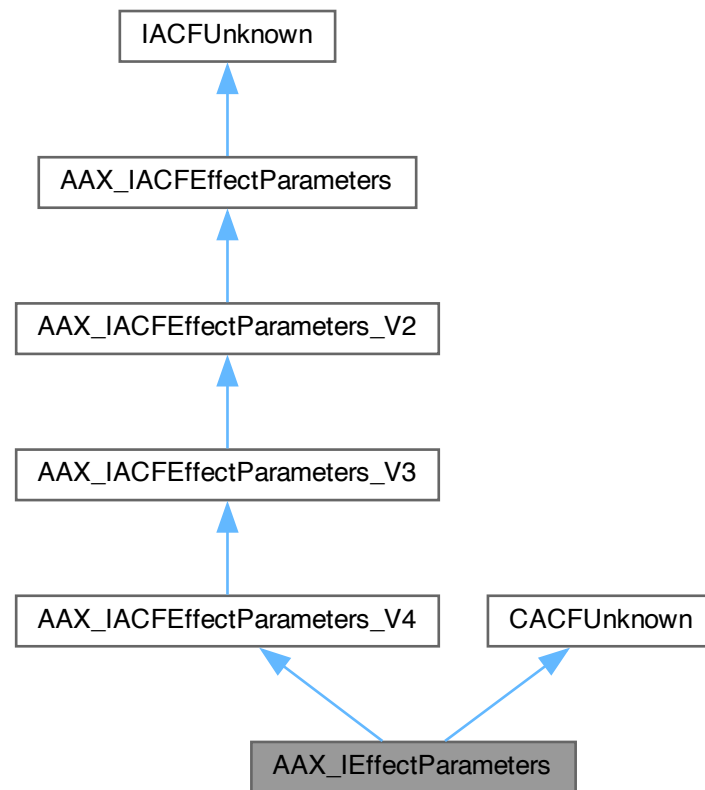
14.111 AAX_IEffectParameters Class Reference

```
#include <AAX_IEffectParameters.h>
```

Inheritance diagram for AAX_IEffectParameters:



Collaboration diagram for AAX_IEffectParameters:



14.111.1 Description

The interface for an AAX Plug-in's data model.

:Implemented by the Plug-In

The interface for an instance of a plug-in's data model. A plug-in's implementation of this interface is responsible for creating the plug-in's set of parameters and for defining how the plug-in will respond when these parameters are changed via control updates or preset loads. In order for information to be routed from the plug-in's data model to its algorithm, the parameters that are created here must be registered with the host in the plug-in's [Description callback](#).

At [initialization](#), the host provides this interface with a reference to [AAX_IController](#), which provides access from the data model back to the host. This reference provides a means of querying information from the host such as stem format or sample rate, and is also responsible for communication between the data model and the plug-in's (decoupled) algorithm. See [Real-time algorithm callback](#).

You will most likely inherit your implementation of this interface from [AAX_CEffectParameters](#), a default implementation that provides basic data model functionality such as adding custom parameters, setting control values, restoring state, generating coefficients, etc., which you can override and customize as needed.

The following tags appear in the descriptions for methods of this class and its derived classes:

- **CALL**: Components in the plug-in should call this method to get / set data in the data model.

Note

- This class always inherits from the latest version of the interface and thus requires any subclass to implement all the methods in the latest version of the interface. The current version of [AAX_CEffectParameters](#) provides a convenient default implementation for all methods in the latest interface.
- Except where noted otherwise, the parameter values referenced by the methods in this interface are normalized values. See [Parameter Manager](#) for more information.

Legacy Porting Notes In the legacy plug-in SDK, these methods were found in CProcess and CEffectParameters. For additional CProcess methods, see [AAX_IEffectGUI](#).

14.111.2 Related classes

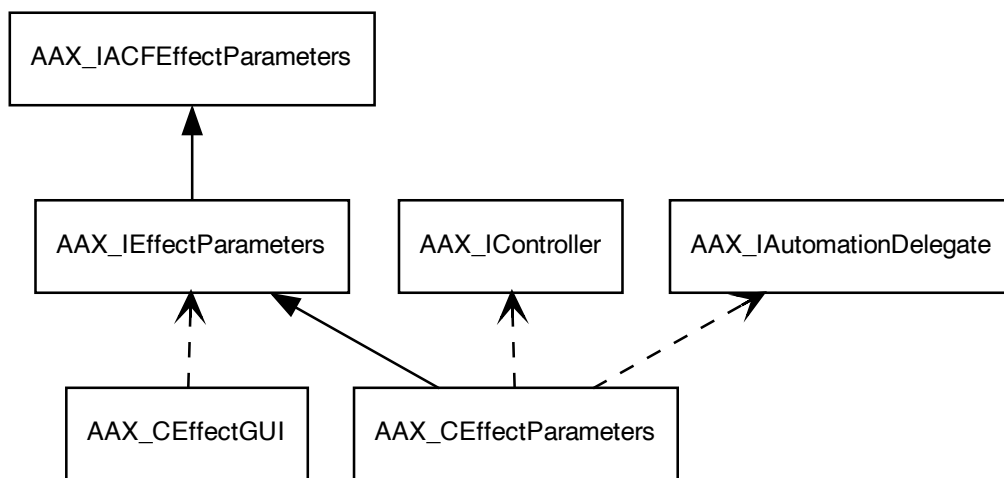


Figure 14.3 Classes related to AAX_IEffectParameters by inheritance or composition

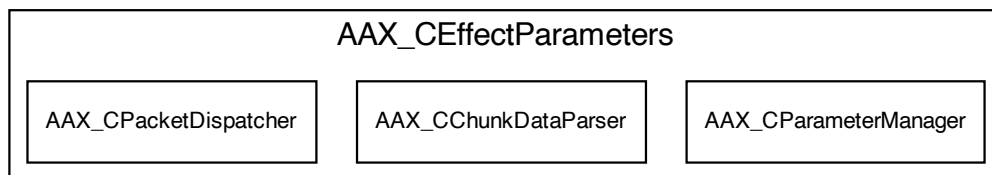


Figure 14.4 Classes owned as member objects of AAX_CEffectParameters

Public Member Functions

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfiID](#) &riid
- [AAX_DELETE](#) (AAX_IEffectParameters &operator=(const AAX_IEffectParameters &))

Public Member Functions inherited from [AAX_IACFEffEffectParameters_V4](#)

- virtual [AAX_Result](#) [UpdatePageTable](#) (uint32_t inTableType, int32_t inTablePageSize, [IACFUnknown](#) *iHostUnknown, [IACFUnknown](#) *ioPageTableUnknown) const =0

Allow the plug-in to update its page tables.

Public Member Functions inherited from [AAX_IACFEffEffectParameters_V3](#)

- virtual [AAX_Result](#) [GetCurveDataMeterIds](#) ([AAX_CTypeID](#) iCurveType, uint32_t *oXMeterId, uint32_t *oYMeterId) const =0

Indicates which meters correspond to the X and Y axes of the EQ or Dynamics graph.

- virtual [AAX_Result](#) [GetCurveDataDisplayRange](#) ([AAX_CTypeID](#) iCurveType, float *oXMin, float *oXMax, float *oYMin, float *oYMax) const =0

Determines the range of the graph shown by the plug-in.

Public Member Functions inherited from [AAX_IACFEffEffectParameters_V2](#)

- virtual [AAX_Result](#) [RenderAudio_Hybrid](#) ([AAX_SHybridRenderInfo](#) *ioRenderInfo)=0

Hybrid audio render function.

- virtual [AAX_Result](#) [UpdateMIDINodes](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_CMidiPacket](#) &iPacket)=0

MIDI update callback.

- virtual [AAX_Result](#) [UpdateControlMIDINodes](#) ([AAX_CTypeID](#) nodeID, [AAX_CMidiPacket](#) &iPacket)=0

MIDI update callback for control MIDI nodes.

Public Member Functions inherited from [AAX_IACFEffEffectParameters](#)

- virtual [AAX_Result](#) [Initialize](#) ([IACFUnknown](#) *iController)=0

Main data model initialization. Called when plug-in instance is first instantiated.

- virtual [AAX_Result](#) [Uninitialize](#) ()=0

Main data model uninitialization.

- virtual [AAX_Result](#) [NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0

Notification Hook.

- virtual [AAX_Result](#) [GetNumberOfParameters](#) (int32_t *oNumControls) const =0

CALL: Retrieves the total number of plug-in parameters.

- virtual [AAX_Result](#) [GetMasterBypassParameter](#) ([AAX_IString](#) *oIDString) const =0

CALL: Retrieves the ID of the plug-in's Master Bypass parameter.

- virtual [AAX_Result](#) [GetParameterIsAutomatable](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *oAutomatable) const =0

CALL: Retrieves information about a parameter's automatable status.

- virtual [AAX_Result](#) [GetParameterNumberOfSteps](#) ([AAX_CParamID](#) iParameterID, int32_t *oNumSteps) const =0

CALL: Retrieves the number of discrete steps for a parameter.

- virtual [AAX_Result](#) [GetParameterName](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName) const =0

CALL: Retrieves the full name for a parameter.

- virtual [AAX_Result](#) [GetParameterNameOfLength](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oName, [int32_t](#) iNameLength) const =0
CALL: Retrieves an abbreviated name for a parameter.
- virtual [AAX_Result](#) [GetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double *oValue) const =0
CALL: Retrieves default value of a parameter.
- virtual [AAX_Result](#) [SetParameterDefaultNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
CALL: Sets the default value of a parameter.
- virtual [AAX_Result](#) [GetParameterType](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterType](#) *oParameterType) const =0
CALL: Retrieves the type of a parameter.
- virtual [AAX_Result](#) [GetParameterOrientation](#) ([AAX_CParamID](#) iParameterID, [AAX_EParameterOrientation](#) *oParameterOrientation) const =0
CALL: Retrieves the orientation that should be applied to a parameter's controls.
- virtual [AAX_Result](#) [GetParameter](#) ([AAX_CParamID](#) iParameterID, [AAX_IParameter](#) **oParameter)=0
CALL: Retrieves an arbitrary setting within a parameter.
- virtual [AAX_Result](#) [GetParameterIndex](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) *oControllIndex) const =0
CALL: Retrieves the index of a parameter.
- virtual [AAX_Result](#) [GetParameterIDFromIndex](#) ([int32_t](#) iControllIndex, [AAX_IString](#) *oParameterIDString) const =0
CALL: Retrieves the ID of a parameter.
- virtual [AAX_Result](#) [GetParameterValueInfo](#) ([AAX_CParamID](#) iParameterID, [int32_t](#) iSelector, [int32_t](#) *oValue) const =0
CALL: Retrieves a property of a parameter.

- virtual [AAX_Result](#) [GetParameterValueFromString](#) ([AAX_CParamID](#) iParameterID, double *oValue, const [AAX_IString](#) &iValueString) const =0
CALL: Converts a value string to a value.
- virtual [AAX_Result](#) [GetParameterStringFromValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const =0
CALL: Converts a normalized parameter value into a string representing its corresponding real value.
- virtual [AAX_Result](#) [GetParameterValueString](#) ([AAX_CParamID](#) iParameterID, [AAX_IString](#) *oValueString, [int32_t](#) iMaxLength) const =0
CALL: Retrieves the value string associated with a parameter's current value.
- virtual [AAX_Result](#) [GetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double *oValuePtr) const =0
CALL: Retrieves a parameter's current value.
- virtual [AAX_Result](#) [SetParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
CALL: Sets the specified parameter to a new value.
- virtual [AAX_Result](#) [SetParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0
CALL: Sets the specified parameter to a new value relative to its current value.

- virtual [AAX_Result](#) [TouchParameter](#) ([AAX_CParamID](#) iParameterID)=0
"Touches" (locks) a parameter in the automation system to a particular control in preparation for updates
- virtual [AAX_Result](#) [ReleaseParameter](#) ([AAX_CParamID](#) iParameterID)=0
Releases a parameter from a "touched" state.
- virtual [AAX_Result](#) [UpdateParameterTouch](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) iTouchState)=0
Sets a "touched" state on a parameter.

- virtual [AAX_Result](#) [UpdateParameterNormalizedValue](#) ([AAX_CParamID](#) iParameterID, double iValue, [AAX_EUpdateSource](#) iSource)=0

- Updates a single parameter's state to its current value.*

 - virtual [AAX_Result UpdateParameterNormalizedRelative](#) ([AAX_CParamID](#) iParameterID, double iValue)=0

Updates a single parameter's state to its current value, as a difference with the parameter's previous value.
- virtual [AAX_Result GenerateCoefficients](#) ()=0

Generates and dispatches new coefficient packets.
- virtual [AAX_Result ResetFieldData](#) ([AAX_CFieldIndex](#) inFieldIndex, void *oData, uint32_t inDataSize) const =0

Called by the host to reset a private data field in the plug-in's algorithm.
- virtual [AAX_Result GetNumberOfChunks](#) (int32_t *oNumChunks) const =0

Retrieves the number of chunks used by this plug-in.
- virtual [AAX_Result GetChunkIDFromIndex](#) (int32_t iIndex, [AAX_CTypeID](#) *oChunkID) const =0

Retrieves the ID associated with a chunk index.
- virtual [AAX_Result GetChunkSize](#) ([AAX_CTypeID](#) iChunkID, uint32_t *oSize) const =0

Get the size of the data structure that can hold all of a chunk's information.
- virtual [AAX_Result GetChunk](#) ([AAX_CTypeID](#) iChunkID, [AAX_SPlugInChunk](#) *oChunk) const =0

Fills a block of data with chunk information representing the plug-in's current state.
- virtual [AAX_Result SetChunk](#) ([AAX_CTypeID](#) iChunkID, const [AAX_SPlugInChunk](#) *iChunk)=0

Restores a set of plug-in parameters based on chunk information.
- virtual [AAX_Result CompareActiveChunk](#) (const [AAX_SPlugInChunk](#) *iChunkP, [AAX_CBoolean](#) *oIsEqual) const =0

Determine if a chunk represents settings that are equivalent to the plug-in's current state.
- virtual [AAX_Result GetNumberOfChanges](#) (int32_t *oNumChanges) const =0

Retrieves the number of parameter changes made since the plug-in's creation.
- virtual [AAX_Result TimerWakeup](#) ()=0

Periodic wakeup callback for idle-time operations.
- virtual [AAX_Result GetCurveData](#) ([AAX_CTypeID](#) iCurveType, const float *iValues, uint32_t iNumValues, float *oValues) const =0

Generate a set of output values based on a set of given input values.
- virtual [AAX_Result GetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, void *oData, uint32_t *oDataWritten) const =0

An optional interface hook for getting custom data from another module.
- virtual [AAX_Result SetCustomData](#) ([AAX_CTypeID](#) iDataBlockID, uint32_t inDataSize, const void *iData)=0

An optional interface hook for setting custom data for use by another module.
- virtual [AAX_Result DoMIDITransfers](#) ()=0

MIDI update callback.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0

Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0

Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0

Decrements reference count.

Public Attributes

- void **ppvObjOut [override](#)

14.111.3 Member Function Documentation

14.111.3.1 ACF_DECLARE_STANDARD_UNKNOWN()

```
AAX_IEffectParameters::ACF_DECLARE_STANDARD_UNKNOWN ( ) const &
```

14.111.3.2 AAX_DELETE()

```
AAX_IEffectParameters::AAX_DELETE (
    AAX_IEffectParameters & operator = (const AAX_IEffectParameters &) )
```

14.111.4 Member Data Documentation

14.111.4.1 override

```
void** ppvObjOut AAX_IEffectParameters::override
```

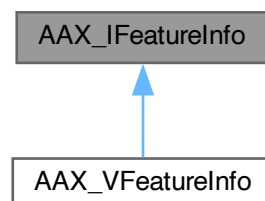
The documentation for this class was generated from the following file:

- [AAX_IEffectParameters.h](#)

14.112 AAX_IFeatureInfo Class Reference

```
#include <AAX_IFeatureInfo.h>
```

Inheritance diagram for AAX_IFeatureInfo:



Public Member Functions

- virtual [~AAX_IFeatureInfo](#) ()
- virtual [AAX_Result](#) [SupportLevel](#) ([AAX_ESupportLevel](#) &oSupportLevel) const =0
- virtual const [AAX_IPropertyMap](#) * [AcquireProperties](#) () const =0
- virtual const [AAX_Feature_UID](#) & [ID](#) () const =0

14.112.1 Constructor & Destructor Documentation

14.112.1.1 ~AAX_IFeatureInfo()

```
virtual AAX_IFeatureInfo::~~AAX_IFeatureInfo ( ) [inline], [virtual]
```

14.112.2 Member Function Documentation

14.112.2.1 SupportLevel()

```
virtual AAX\_Result AAX_IFeatureInfo::SupportLevel (
    AAX\_ESupportLevel & oSupportLevel ) const [pure virtual]
```

Determine the level of support for this feature by the host

Note

The host will not provide an underlying [AAX_IACFFeatureInfo](#) interface for features which it does not recognize at all, resulting in a [AAX_ERROR_NULL_OBJECT](#) error code

Implemented in [AAX_VFeatureInfo](#).

14.112.2.2 AcquireProperties()

```
virtual const AAX\_IPropertyMap * AAX_IFeatureInfo::AcquireProperties ( ) const [pure virtual]
```

Additional properties providing details of the feature support

See the feature's UID for documentation of which features provide additional properties

Ownership of the returned object is passed to the caller; the caller is responsible for destroying the object, e.g. by capturing the returned object in a smart pointer.

```
// AAX_IFeatureInfo* featureInfo
std::unique_ptr<const AAX_IPropertyMap> featurePropertiesPtr(featureInfo->AcquireProperties());
```

Returns

An [AAX_IPropertyMap](#) interface with access to the host's properties for this feature.

NULL if the desired feature was not found or if an error occurred

Note

May return an [AAX_IPropertyMap](#) object with limited method support, which would return an error such as [AAX_ERROR_NULL_OBJECT](#) or [AAX_ERROR_UNIMPLEMENTED](#) to interface calls.

Implemented in [AAX_VFeatureInfo](#).

14.112.2.3 ID()

```
virtual const AAX\_Feature\_UID & AAX_IFeatureInfo::ID ( ) const [pure virtual]
```

Returns the ID of the feature which this object represents

Implemented in [AAX_VFeatureInfo](#).

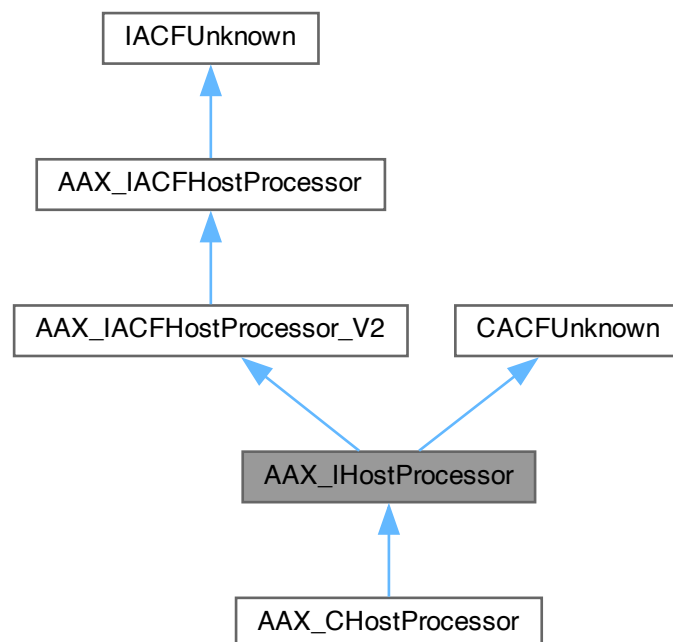
The documentation for this class was generated from the following file:

- [AAX_IFeatureInfo.h](#)

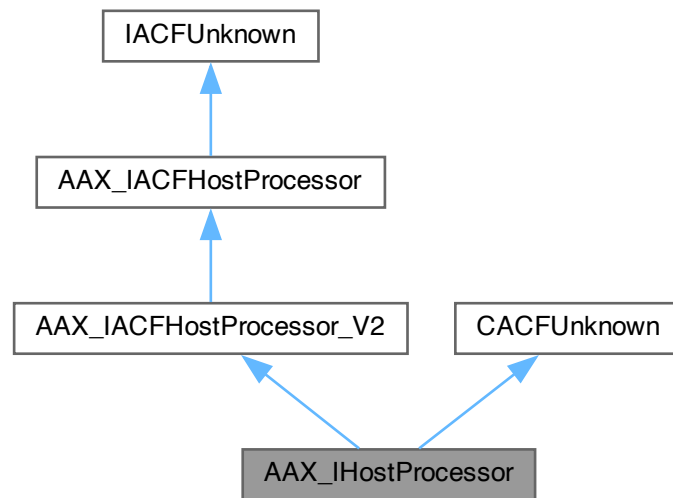
14.113 AAX_IHostProcessor Class Reference

```
#include <AAX_IHostProcessor.h>
```

Inheritance diagram for AAX_IHostProcessor:



Collaboration diagram for AAX_IHostProcessor:



14.113.1 Description

Base class for the host processor interface.

:Implemented by the Plug-In

Note

This class always inherits from the latest version of the interface and thus requires any subclass to implement all the methods in the latest version of the interface. Most plug-ins will inherit from the [AAX_CHostProcessor](#) convenience class.

Public Member Functions

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acflID](#) &riid
- [AAX_DELETE](#) ([AAX_IHostProcessor](#) &operator=(const [AAX_IHostProcessor](#) &))

Public Member Functions inherited from [AAX_IACFHostProcessor_V2](#)

- virtual [AAX_Result](#) [GetClipNameSuffix](#) (int32_t inMaxLength, [AAX_IString](#) *outString) const =0
Called by host application to retrieve a custom string to be appended to the clip name.

Public Member Functions inherited from [AAX_IACFHostProcessor](#)

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iController)=0
Host Processor initialization.
- virtual [AAX_Result Uninitialize](#) ()=0
Host Processor teardown.
- virtual [AAX_Result InitOutputBounds](#) (int64_t iSrcStart, int64_t iSrcEnd, int64_t *oDstStart, int64_t *oDstEnd)=0
Sets the processing region.
- virtual [AAX_Result SetLocation](#) (int64_t iSample)=0
Updates the relative sample location of the current processing frame.
- virtual [AAX_Result RenderAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, float *const iAudioOuts[], int32_t iAudioOutCount, int32_t *ioWindowSize)=0
Perform the signal processing.
- virtual [AAX_Result PreRender](#) (int32_t inAudioInCount, int32_t iAudioOutCount, int32_t iWindowSize)=0
Invoked right before the start of a Preview or Render pass.
- virtual [AAX_Result PostRender](#) ()=0
Invoked at the end of a Render pass.
- virtual [AAX_Result AnalyzeAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int32_t *ioWindowSize)=0
Override this method if the plug-in needs to analyze the audio prior to a Render pass.
- virtual [AAX_Result PreAnalyze](#) (int32_t inAudioInCount, int32_t iWindowSize)=0
Invoked right before the start of an Analysis pass.
- virtual [AAX_Result PostAnalyze](#) ()=0
Invoked at the end of an Analysis pass.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Public Attributes

- void **ppvObjOut [override](#)

14.113.2 Member Function Documentation**14.113.2.1 ACF_DECLARE_STANDARD_UNKNOWN()**

```
AAX_IHostProcessor::ACF_DECLARE_STANDARD_UNKNOWN ( ) const &
```

14.113.2.2 AAX_DELETE()

```
AAX_IHostProcessor::AAX_DELETE (
    AAX_IHostProcessor & operator = (const AAX_IHostProcessor &) )
```

14.113.3 Member Data Documentation

14.113.3.1 override

```
void** ppvObjOut AAX_IHostProcessor::override
```

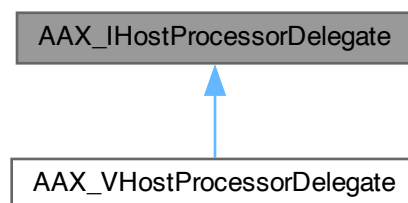
The documentation for this class was generated from the following file:

- [AAX_IHostProcessor.h](#)

14.114 AAX_IHostProcessorDelegate Class Reference

```
#include <AAX_IHostProcessorDelegate.h>
```

Inheritance diagram for AAX_IHostProcessorDelegate:



14.114.1 Description

Versioned interface for host methods specific to offline processing.

:Implemented by the AAX Host

The host provides a host processor delegate to a plug-in's [host processor](#) object at initialization. The host processor object may make calls to this object to get information about the current render pass or to affect the plug-in's offline processing behavior.

Public Member Functions

- virtual [~AAX_IHostProcessorDelegate](#) ()
- virtual [AAX_Result GetAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int64_t inLocation, int32_t *ioNumSamples)=0
CALL: Randomly access audio from the timeline.
- virtual int32_t [GetSideChainInputNum](#) ()=0
CALL: Returns the index of the side chain input buffer.
- virtual [AAX_Result ForceAnalyze](#) ()=0
CALL: Request an analysis pass.
- virtual [AAX_Result ForceProcess](#) ()=0
CALL: Request a process pass.

14.114.2 Constructor & Destructor Documentation

14.114.2.1 ~AAX_IHostProcessorDelegate()

```
virtual AAX_IHostProcessorDelegate::~~AAX_IHostProcessorDelegate ( ) [inline], [virtual]
```

14.114.3 Member Function Documentation

14.114.3.1 GetAudio()

```
virtual AAX_Result AAX_IHostProcessorDelegate::GetAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    int64_t inLocation,
    int32_t * ioNumSamples ) [pure virtual]
```

CALL: Randomly access audio from the timeline.

Called from within [AAX_IHostProcessor::RenderAudio\(\)](#), this method fills a buffer of samples with randomly-accessed data from the current input processing region on the timeline, including any extra samples such as processing "handles".

Note

Plug-ins that use this feature must set [AAX_eProperty_UsesRandomAccess](#) to `true`

It is not possible to retrieve samples from outside of the current input processing region

Always check the return value of this method before using the randomly-accessed samples

Parameters

in	<i>inAudioIns</i>	Timeline audio buffer(s). This must be set to <i>inAudioIns</i> from AAX_IHostProcessor::RenderAudio()
in	<i>inAudioInCount</i>	Number of buffers in <i>inAudioIns</i> . This must be set to <i>inAudioInCount</i> from AAX_IHostProcessor::RenderAudio()
in	<i>inLocation</i>	A sample location relative to the beginning of the currently processed region, e.g. a value of 0 corresponds to the timeline location returned by AAX_CHostProcessor::GetSrcStart()
in, out	<i>ioNumSamples</i>	<ul style="list-style-type: none"> • Input: The maximum number of samples to read. • Output: The actual number of samples that were read from the timeline

Implemented in [AAX_VHostProcessorDelegate](#).

14.114.3.2 GetSideChainInputNum()

```
virtual int32_t AAX_IHostProcessorDelegate::GetSideChainInputNum ( ) [pure virtual]
```

CALL: Returns the index of the side chain input buffer.

Called from within [AAX_IHostProcessor::RenderAudio\(\)](#), this method returns the index of the side chain input sample buffer within *inAudioIns*.

Implemented in [AAX_VHostProcessorDelegate](#).

14.114.3.3 ForceAnalyze()

```
virtual AAX_Result AAX_IHostProcessorDelegate::ForceAnalyze ( ) [pure virtual]
```

CALL: Request an analysis pass.

Call this method to request an analysis pass from within the plug-in. Most plug-ins should rely on the host to trigger analysis passes when appropriate. However, plug-ins that require an analysis pass a) outside of the context of host-driven render or analysis, or b) when internal plug-in data changes may need to call [ForceAnalyze\(\)](#).

Implemented in [AAX_VHostProcessorDelegate](#).

14.114.3.4 ForceProcess()

```
virtual AAX\_Result AAX_IHostProcessorDelegate::ForceProcess ( ) [pure virtual]
```

CALL: Request a process pass.

Call this method to request a process pass from within the plug-in. If [AAX_eProperty_RequiresAnalysis](#) is defined, the resulting process pass will be preceded by an analysis pass. This method should only be used in rare circumstances by plug-ins that must launch processing outside of the normal host AudioSuite workflow.

Implemented in [AAX_VHostProcessorDelegate](#).

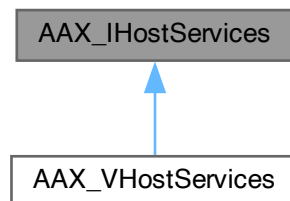
The documentation for this class was generated from the following file:

- [AAX_IHostProcessorDelegate.h](#)

14.115 AAX_IHostServices Class Reference

```
#include <AAX_IHostServices.h>
```

Inheritance diagram for AAX_IHostServices:



14.115.1 Description

Interface to diagnostic and debugging services provided by the AAX host.

:Implemented by the AAX Host

See also

[AAX_IACFHostServices](#)

Public Member Functions

- virtual [~AAX_IHostServices](#) ()
- virtual [AAX_Result HandleAssertFailure](#) (const char *iFile, int32_t iLine, const char *iNote, int32_t iFlags) const =0
Handle an assertion failure.
- virtual [AAX_Result Trace](#) (int32_t iPriority, const char *iMessage) const =0
Log a trace message.
- virtual [AAX_Result StackTrace](#) (int32_t iTracePriority, int32_t iStackTracePriority, const char *iMessage) const =0
Log a trace message or a stack trace.

14.115.2 Constructor & Destructor Documentation

14.115.2.1 ~AAX_IHostServices()

```
virtual AAX_IHostServices::~~AAX_IHostServices ( ) [inline], [virtual]
```

14.115.3 Member Function Documentation

14.115.3.1 HandleAssertFailure()

```
virtual AAX\_Result AAX_IHostServices::HandleAssertFailure (
    const char * iFile,
    int32_t iLine,
    const char * iNote,
    int32_t iFlags ) const [pure virtual]
```

Handle an assertion failure.

Use this method to delegate assertion failure handling to the host

Use `iFlags` to request that specific behavior be included when handling the failure. This request may not be fulfilled by the host, and absence of a flag does not preclude the host from using that behavior when handling the failure.

Parameters

in	<i>iFile</i>	The name of the file containing the assert check. Usually <code>__FILE__</code>
in	<i>iLine</i>	The line number of the assert check. Usually <code>__LINE__</code>
in	<i>iNote</i>	Text to display related to the assert. Usually the condition which failed
in	<i>iFlags</i>	Bitfield of AAX_EAssertFlags to request specific handling behavior

Implemented in [AAX_VHostServices](#).

14.115.3.2 Trace()

```
virtual AAX_Result AAX_IHostServices::Trace (
    int32_t iPriority,
    const char * iMessage ) const [pure virtual]
```

Log a trace message.

Parameters

in	<i>iPriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iMessage</i>	Message string to log

Implemented in [AAX_VHostServices](#).

14.115.3.3 StackTrace()

```
virtual AAX_Result AAX_IHostServices::StackTrace (
    int32_t iTracePriority,
    int32_t iStackTracePriority,
    const char * iMessage ) const [pure virtual]
```

Log a trace message or a stack trace.

If the logging output filtering is set to include logs with *iStackTracePriority* then both the logging message and a stack trace will be emitted, regardless of *iTracePriority*.

If the logging output filtering is set to include logs with *iTracePriority* but to exclude logs with *iStackTracePriority* then this will emit a normal log with no stack trace.

Parameters

in	<i>iTracePriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iStackTracePriority</i>	Priority of the stack trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iMessage</i>	Message string to log

Implemented in [AAX_VHostServices](#).

The documentation for this class was generated from the following file:

- [AAX_IHostServices.h](#)

14.116 AAX_IMIDIMessageInfoDelegate Class Reference

Public Member Functions

- virtual [~AAX_IMIDIMessageInfoDelegate](#) ()
- virtual uint32_t [Mask](#) () const =0
- virtual uint32_t [Length](#) () const =0
- virtual [AAX_CString ToString](#) (uint32_t inLength, const uint8_t *inData) const =0
- virtual bool [Accepts](#) (uint32_t inLength, const uint8_t *inData) const

Protected Member Functions

- bool [Accepts_ExactStatus](#) (uint32_t inLength, const uint8_t *inData) const

Static Protected Member Functions

- static void [ToString_AppendNumber](#) (const char *inLabel, int32_t inData, [AAX_CString](#) &outString)
- static void [ToString_AppendCStr](#) (const char *inLabel, const char *inCStr, [AAX_CString](#) &outString)
- static void [ToString_AppendByteRange](#) (const char *inLabel, const uint8_t *inData, int32_t inNumBytes, [AAX_CString](#) &outString)
- static void [ToString_AppendValid](#) (bool inCheck, [AAX_CString](#) &outString)

14.116.1 Constructor & Destructor Documentation

14.116.1.1 ~AAX_IMIDIMessageInfoDelegate()

```
virtual AAX_IMIDIMessageInfoDelegate::~~AAX_IMIDIMessageInfoDelegate ( ) [inline], [virtual]
```

14.116.2 Member Function Documentation

14.116.2.1 Mask()

```
virtual uint32_t AAX_IMIDIMessageInfoDelegate::Mask ( ) const [pure virtual]
```

Referenced by [Accepts\(\)](#), and [Accepts_ExactStatus\(\)](#).

Here is the caller graph for this function:



14.116.2.2 Length()

```
virtual uint32_t AAX_IMIDMessageInfoDelegate::Length ( ) const [pure virtual]
```

Referenced by [Accepts\(\)](#).

Here is the caller graph for this function:



14.116.2.3 ToString()

```
virtual AAX_CString AAX_IMIDMessageInfoDelegate::ToString (
    uint32_t inLength,
    const uint8_t * inData ) const [pure virtual]
```

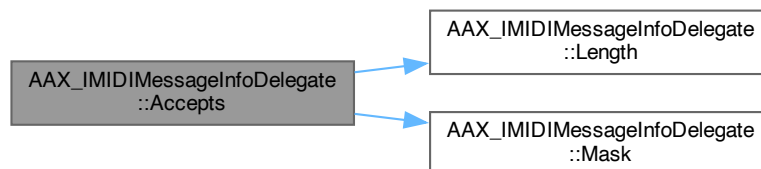
14.116.2.4 Accepts()

```
virtual bool AAX_IMIDMessageInfoDelegate::Accepts (
    uint32_t inLength,
    const uint8_t * inData ) const [inline], [virtual]
```

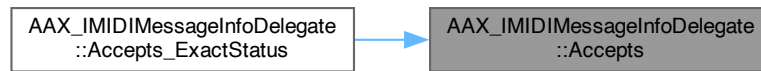
References [Length\(\)](#), and [Mask\(\)](#).

Referenced by [Accepts_ExactStatus\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

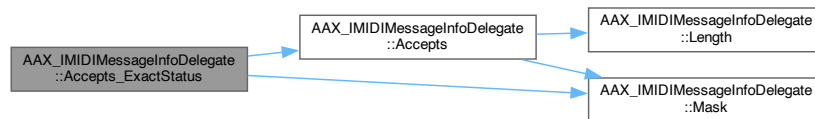


14.116.2.5 Accepts_ExactStatus()

```
bool AAX_IMIDIMessageInfoDelegate::Accepts_ExactStatus (
    uint32_t inLength,
    const uint8_t * inData ) const [inline], [protected]
```

References [Accepts\(\)](#), and [Mask\(\)](#).

Here is the call graph for this function:



14.116.2.6 ToString_AppendNumber()

```
static void AAX_IMIDIMessageInfoDelegate::ToString_AppendNumber (
    const char * inLabel,
    int32_t inData,
    AAX_CString & outString ) [inline], [static], [protected]
```

References [AAX_CString::AppendNumber\(\)](#).

Here is the call graph for this function:



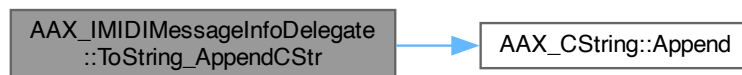
14.116.2.7 ToString_AppendCStr()

```
static void AAX_IMIDIMessageInfoDelegate::ToString_AppendCStr (
    const char * inLabel,
    const char * inCStr,
    AAX_CString & outString ) [inline], [static], [protected]
```

References [AAX_CString::Append\(\)](#).

Referenced by [ToString_AppendValid\(\)](#).

Here is the call graph for this function:



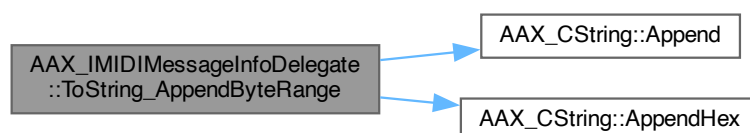
Here is the caller graph for this function:

**14.116.2.8 ToString_AppendByteRange()**

```
static void AAX_IMIDIMessageInfoDelegate::ToString_AppendByteRange (
    const char * inLabel,
    const uint8_t * inData,
    int32_t inNumBytes,
    AAX_CString & outString ) [inline], [static], [protected]
```

References [AAX_CString::Append\(\)](#), and [AAX_CString::AppendHex\(\)](#).

Here is the call graph for this function:

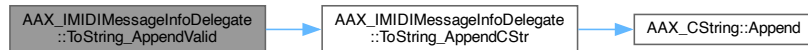


14.116.2.9 ToString_AppendValid()

```
static void AAX_IMIDIMessageInfoDelegate::ToString_AppendValid (
    bool inCheck,
    AAX_CString & outString ) [inline], [static], [protected]
```

References [ToString_AppendCStr\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [AAX_MIDILogging.cpp](#)

14.117 AAX_IMIDINode Class Reference

```
#include <AAX_IMIDINode.h>
```

14.117.1 Description

Interface for accessing information in a MIDI node.

:Implemented by the AAX Host

See also

[AAX_IComponentDescriptor::AddMIDINode](#)

Public Member Functions

- virtual `~AAX_IMIDINode ()`
- virtual `AAX_CMidiStream * GetNodeBuffer ()=0`
Returns a MIDI stream data structure.
- virtual `AAX_Result PostMIDIPacket (AAX_CMidiPacket *packet)=0`
Posts an AAX_CMidiPacket to an output MIDI node.
- virtual `AAX_ITransport * GetTransport ()=0`
Returns a transport object.

14.117.2 Constructor & Destructor Documentation

14.117.2.1 ~AAX_IMIDINode()

```
virtual AAX_IMIDINode::~~AAX_IMIDINode ( ) [inline], [virtual]
```

14.117.3 Member Function Documentation

14.117.3.1 GetNodeBuffer()

```
virtual AAX_CMidiStream * AAX_IMIDINode::GetNodeBuffer ( ) [pure virtual]
```

Returns a MIDI stream data structure.

14.117.3.2 PostMIDIPacket()

```
virtual AAX_Result AAX_IMIDINode::PostMIDIPacket (
    AAX_CMidiPacket * packet ) [pure virtual]
```

Posts an [AAX_CMidiPacket](#) to an output MIDI node.

Host Compatibility Notes Pro Tools supports the following MIDI events from plug-ins:

- NoteOn
- NoteOff
- Pitch bend
- Polyphonic key pressure
- Bank select (controller #0)
- Program change (no bank)
- Channel pressure

Parameters

<i>in</i>	<i>packet</i>	The MIDI packet to be pushed to a MIDI output node
-----------	---------------	--

14.117.3.3 GetTransport()

```
virtual AAX\_ITransport * AAX_IMIDINode::GetTransport ( ) [pure virtual]
```

Returns a transport object.

Warning

The returned interface is not versioned. Calling a method on this interface that is not supported by the host will result in undefined behavior, usually a crash. You must either check the host version before using this interface or limit the use of this interface to [V1 Transport interface](#) methods.

Wherever possible, use a versioned Transport object such as the one created in [AAX_CEffectParameters::Initialize\(\)](#) rather than this unversioned interface.

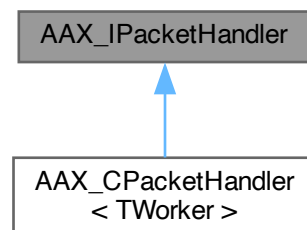
The documentation for this class was generated from the following file:

- [AAX_IMIDINode.h](#)

14.118 AAX_IPacketHandler Struct Reference

```
#include <AAX_CPacketDispatcher.h>
```

Inheritance diagram for AAX_IPacketHandler:



14.118.1 Description

Callback container used by [AAX_CPacketDispatcher](#).

Public Member Functions

- virtual `~AAX_IPacketHandler ()`
- virtual `AAX_IPacketHandler * Clone () const =0`
- virtual `AAX_Result Call (AAX_CParamID inParamID, AAX_CPacket &ioPacket) const =0`

14.118.2 Constructor & Destructor Documentation

14.118.2.1 ~AAX_IPacketHandler()

```
virtual AAX_IPacketHandler::~~AAX_IPacketHandler ( ) [inline], [virtual]
```

14.118.3 Member Function Documentation

14.118.3.1 Clone()

```
virtual AAX_IPacketHandler * AAX_IPacketHandler::Clone ( ) const [pure virtual]
```

Implemented in [AAX_CPacketHandler< TWorker >](#).

14.118.3.2 Call()

```
virtual AAX_Result AAX_IPacketHandler::Call (
    AAX_CParamID inParamID,
    AAX_CPacket & ioPacket ) const [pure virtual]
```

Implemented in [AAX_CPacketHandler< TWorker >](#).

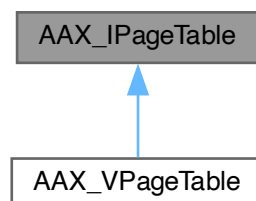
The documentation for this struct was generated from the following file:

- [AAX_CPacketDispatcher.h](#)

14.119 AAX_IPageTable Class Reference

```
#include <AAX_IPageTable.h>
```

Inheritance diagram for AAX_IPageTable:



14.119.1 Description

Interface to the host's representation of a plug-in instance's page table.

See also

[AAX_IEffectParameters::UpdatePageTable\(\)](#)

Public Member Functions

- virtual [~AAX_IPageTable](#) ()
Virtual destructor.
- virtual [AAX_Result Clear](#) ()=0
Clears all parameter mappings from the table.
- virtual [AAX_Result Empty](#) ([AAX_CBoolean](#) &oEmpty) const =0
Indicates whether the table is empty.
- virtual [AAX_Result GetNumPages](#) (int32_t &oNumPages) const =0
Get the number of pages currently in this table.
- virtual [AAX_Result InsertPage](#) (int32_t iPage)=0
Insert a new empty page before the page at index iPage.
- virtual [AAX_Result RemovePage](#) (int32_t iPage)=0
Remove the page at index iPage.
- virtual [AAX_Result GetNumMappedParameterIDs](#) (int32_t iPage, int32_t &oNumParameterIdentifiers) const =0
Returns the total number of parameter IDs which are mapped to a page.
- virtual [AAX_Result ClearMappedParameter](#) (int32_t iPage, int32_t iIndex)=0
Clear the parameter at a particular index in this table.
- virtual [AAX_Result GetMappedParameterID](#) (int32_t iPage, int32_t iIndex, [AAX_IString](#) &oParameterIdentifier) const =0
Get the parameter identifier which is currently mapped to an index in this table.
- virtual [AAX_Result MapParameterID](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t iPage, int32_t iIndex)=0
Map a parameter to this table.
- virtual [AAX_Result GetNumParametersWithNameVariations](#) (int32_t &oNumParameterIdentifiers) const =0
- virtual [AAX_Result GetNameVariationParameterIDAtIndex](#) (int32_t iIndex, [AAX_IString](#) &oParameterIdentifier) const =0
- virtual [AAX_Result GetNumNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t &oNumVariations) const =0
- virtual [AAX_Result GetParameterNameVariationAtIndex](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t iIndex, [AAX_IString](#) &oNameVariation, int32_t &oLength) const =0
- virtual [AAX_Result GetParameterNameVariationOfLength](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t iLength, [AAX_IString](#) &oNameVariation) const =0
- virtual [AAX_Result ClearParameterNameVariations](#) ()=0
- virtual [AAX_Result ClearNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier)=0
- virtual [AAX_Result SetParameterNameVariation](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, const [AAX_IString](#) &iNameVariation, int32_t iLength)=0

14.119.2 Constructor & Destructor Documentation

14.119.2.1 ~AAX_IPageTable()

```
virtual AAX_IPageTable::~~AAX_IPageTable ( ) [inline], [virtual]
```

Virtual destructor.

Note

This destructor MUST be virtual to prevent memory leaks.

14.119.3 Member Function Documentation

14.119.3.1 Clear()

```
virtual AAX_Result AAX_IPageTable::Clear ( ) [pure virtual]
```

Clears all parameter mappings from the table.

This method does not clear any parameter name variations from the table. For that, use [AAX_IPageTable::ClearParameterNameVariations](#) or [AAX_IPageTable::ClearNameVariationsForParameter](#)()

Implemented in [AAX_VPageTable](#).

14.119.3.2 Empty()

```
virtual AAX_Result AAX_IPageTable::Empty (
    AAX_CBoolean & oEmpty ) const [pure virtual]
```

Indicates whether the table is empty.

A table is empty if it contains no pages. A table which contains pages but no parameter assignments is not empty. A table which has associated parameter name variations but no pages is empty.

Parameters

out	<i>oEmpty</i>	true if this table is empty
-----	---------------	-----------------------------

Implemented in [AAX_VPageTable](#).

14.119.3.3 GetNumPages()

```
virtual AAX_Result AAX_IPageTable::GetNumPages (
    int32_t & oNumPages ) const [pure virtual]
```

Get the number of pages currently in this table.

Parameters

out	<i>oNumPages</i>	The number of pages which are present in the page table. Some pages might not contain any parameter assignments.
-----	------------------	--

Implemented in [AAX_VPageTable](#).

14.119.3.4 InsertPage()

```
virtual AAX_Result AAX_IPageTable::InsertPage (  
    int32_t iPage ) [pure virtual]
```

Insert a new empty page before the page at index *iPage*.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the total number of pages

Parameters

in	<i>iPage</i>	The insertion point page index
----	--------------	--------------------------------

Implemented in [AAX_VPageTable](#).

14.119.3.5 RemovePage()

```
virtual AAX_Result AAX_IPageTable::RemovePage (  
    int32_t iPage ) [pure virtual]
```

Remove the page at index *iPage*.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

Parameters

in	<i>iPage</i>	The target page index
----	--------------	-----------------------

Implemented in [AAX_VPageTable](#).

14.119.3.6 GetNumMappedParameterIDs()

```
virtual AAX_Result AAX_IPageTable::GetNumMappedParameterIDs (
    int32_t iPage,
    int32_t & oNumParameterIdentifiers ) const [pure virtual]
```

Returns the total number of parameter IDs which are mapped to a page.

Note

The number of mapped parameter IDs does not correspond to the actual slot indices of the parameter assignments. For example, a page could have three total parameter assignments with parameters mapped to slots 2, 4, and 6.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

Parameters

in	<i>iPage</i>	The target page index
out	<i>oNumParameterIdentifiers</i>	The number of parameter identifiers which are mapped to the target page

Implemented in [AAX_VPageTable](#).

14.119.3.7 ClearMappedParameter()

```
virtual AAX_Result AAX_IPageTable::ClearMappedParameter (
    int32_t iPage,
    int32_t iIndex ) [pure virtual]
```

Clear the parameter at a particular index in this table.

Returns

[AAX_SUCCESS](#) even if no parameter was mapped at the given index (the index is still clear)

Parameters

in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page

Implemented in [AAX_VPageTable](#).

14.119.3.8 GetMappedParameterID()

```
virtual AAX_Result AAX_IPageTable::GetMappedParameterID (
    int32_t iPage,
    int32_t iIndex,
    AAX_IString & oParameterIdentifier ) const [pure virtual]
```

Get the parameter identifier which is currently mapped to an index in this table.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if no parameter is mapped at the specified page/index location

Parameters

in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page
out	<i>oParameterIdentifier</i>	The identifier used for the mapped parameter in the page table (may be parameter name or ID)

Implemented in [AAX_VPageTable](#).

14.119.3.9 MapParameterID()

```
virtual AAX_Result AAX_IPageTable::MapParameterID (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t iPage,
    int32_t iIndex ) [pure virtual]
```

Map a parameter to this table.

If *iParameterIdentifier* is an empty string then the parameter assignment will be cleared

Returns

[AAX_ERROR_NULL_ARGUMENT](#) if *iParameterIdentifier* is null

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

[AAX_ERROR_INVALID_ARGUMENT](#) if *iIndex* is negative

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter which will be mapped
in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page

Implemented in [AAX_VPageTable](#).

14.119.3.10 GetNumParametersWithNameVariations()

```
virtual AAX_Result AAX_IPageTable::GetNumParametersWithNameVariations (
    int32_t & oNumParameterIdentifiers ) const [pure virtual]
```

Get the number of parameters with name variations defined for the current table type

Provides the number of parameters with `lt;ControlNameVariationslt;` which are explicitly defined for the current page table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNameVariationParameterIDAtIndex\(\)](#)

Parameters

out	<i>oNumParameterIdentifiers</i>	The number of parameters with name variations explicitly associated with the current table type.
-----	---------------------------------	--

Implemented in [AAX_VPageTable](#).

14.119.3.11 GetNameVariationParameterIDAtIndex()

```
virtual AAX_Result AAX_IPageTable::GetNameVariationParameterIDAtIndex (
    int32_t iIndex,
    AAX_IString & oParameterIdentifier ) const [pure virtual]
```

Get the identifier for a parameter with name variations defined for the current table type

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNumParametersWithNameVariations\(\)](#)

Parameters

in	<i>iIndex</i>	The target parameter index within the list of parameters with explicit name variations defined for this table type.
out	<i>oParameterIdentifier</i>	The identifier used for the parameter in the page table name variations list (may be parameter name or ID)

Implemented in [AAX_VPageTable](#).

14.119.3.12 GetNumNameVariationsForParameter()

```
virtual AAX_Result AAX_IPageTable::GetNumNameVariationsForParameter (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t & oNumVariations ) const [pure virtual]
```

Get the number of name variations defined for a parameter

Provides the number of `lt;ControlNameVariationslt;` which are explicitly defined for `iParameterIdentifier` for the current page table type. No fallback logic is used to resolve this to the list of variations which would actually be used for an attached control surface if no explicit variations are defined for the current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetParameterNameVariationAtIndex\(\)](#)

Returns

[AAX_SUCCESS](#) and provides zero to `oNumVariations` if `iParameterIdentifier` is not found

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
out	<i>oNumVariations</i>	The number of name variations which are defined for this parameter and explicitly associated with the current table type.

Implemented in [AAX_VPageTable](#).

14.119.3.13 GetParameterNameVariationAtIndex()

```
virtual AAX_Result AAX_IPageTable::GetParameterNameVariationAtIndex (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t iIndex,
    AAX_IString & oNameVariation,
    int32_t & oLength ) const [pure virtual]
```

Get a parameter name variation from the page table

Only returns `lt;ControlNameVariationslt;` which are explicitly defined for the current page table type. No fallback logic is used to resolve this to the abbreviation which would actually be shown on an attached control surface if no explicit variation is defined for the current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNumNameVariationsForParameter\(\)](#)

See also

- [AAX_IPageTable::GetParameterNameVariationOfLength\(\)](#)

Returns

[AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME](#) if no suitable variation is defined for this table

[AAX_ERROR_ARGUMENT_OUT_OF_RANGE](#) if `iIndex` is out of range

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iIndex</i>	Index of the name variation
out	<i>oNameVariation</i>	The name variation, if one is explicitly defined for this table type
out	<i>oLength</i>	The length value for this name variation. This corresponds to the variation's <code>sz</code> attribute in the page table XML and may be different from the string length of <code>iNameVariation</code> .

Implemented in [AAX_VPageTable](#).

14.119.3.14 GetParameterNameVariationOfLength()

```
virtual AAX_Result AAX_IPageTable::GetParameterNameVariationOfLength (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t iLength,
    AAX_IString & oNameVariation ) const [pure virtual]
```

Get a parameter name variation of a particular length from the page table

Only returns `lt;ControlNameVariationslt;` which are explicitly defined of `iLength` for the current page table type. No fallback logic is used to resolve this to the abbreviation which would actually be shown on an attached control surface if no explicit variation is defined for the specified length or current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetParameterNameVariationAtIndex\(\)](#)

Returns

[AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME](#) if no suitable variation is defined for this table

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iLength</i>	The variation length to check, i.e. the <i>sz</i> attribute for the name variation in the page table XML
out	<i>oNameVariation</i>	The name variation, if one is explicitly defined for this table type and <i>iLength</i>

Implemented in [AAX_VPageTable](#).

14.119.3.15 ClearParameterNameVariations()

```
virtual AAX_Result AAX_IPageTable::ClearParameterNameVariations ( ) [pure virtual]
```

Clears all name variations for the current page table type

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

[AAX_IPageTable::Clear\(\)](#)

[AAX_IPageTable::ClearNameVariationsForParameter\(\)](#)

Implemented in [AAX_VPageTable](#).

14.119.3.16 ClearNameVariationsForParameter()

```
virtual AAX_Result AAX_IPageTable::ClearNameVariationsForParameter (
    AAX_CPageTableParamID iParameterIdentifier ) [pure virtual]
```

Clears all name variations for a single parameter for the current page table type

Warning

This will invalidate the list of parameter name variations indices, i.e. the parameter identifier associated with each index by [AAX_IPageTable::GetNameVariationParameterIDAtIndex\(\)](#)

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

[AAX_IPageTable::Clear\(\)](#)

[AAX_IPageTable::ClearParameterNameVariations\(\)](#)

Returns

[AAX_SUCCESS](#) and provides zero to *oNumVariations* if *iParameterIdentifier* is not found

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
----	-----------------------------	----------------------------------

Implemented in [AAX_VPageTable](#).

14.119.3.17 SetParameterNameVariation()

```
virtual AAX_Result AAX_IPageTable::SetParameterNameVariation (
    AAX_CPageTableParamID iParameterIdentifier,
    const AAX_IString & iNameVariation,
    int32_t iLength ) [pure virtual]
```

Sets a name variation explicitly for the current page table type

This will add a new name variation or overwrite the existing name variation with the same length which is defined for the current table type.

Warning

If no name variation previously existed for this parameter then this will invalidate the list of parameter name variations indices, i.e. the parameter identifier associated with each index by [AAX_IPageTable::GetNameVariationParameterIDat](#)

Note

Normally parameter name variations are only used with the 'PgTL' table type

Returns

AAX_ERROR_INVALID_ARGUMENT if *iNameVariation* is empty or if *iLength* is less than zero

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iNameVariation</i>	The new parameter name variation
in	<i>iLength</i>	The length value for this name variation. This corresponds to the variation's <i>sz</i> attribute in the page table XML and is not required to match the length of <i>iNameVariation</i> .

Implemented in [AAX_VPageTable](#).

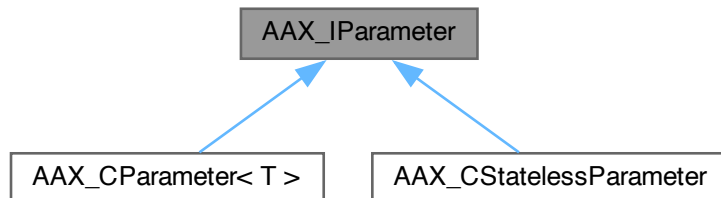
The documentation for this class was generated from the following file:

- [AAX_IPageTable.h](#)

14.120 AAX_IPParameter Class Reference

```
#include <AAX_IPParameter.h>
```

Inheritance diagram for AAX_IPParameter:



14.120.1 Description

The base interface for all normalizable plug-in parameters.

:Internal to the AAX SDK

This class is an outside interface for an arbitrarily typed parameter. The subclasses of this generic interface hold the parameter's state and conversion functionality.

Note

This class is *not* part of the AAX ABI and must not be passed between the plug-in and the host. Version checking is recommended when passing references to this interface between plug-in modules (e.g. between the data model and the GUI)

Public Member Functions

- virtual `~AAX_IPParameter ()`
Virtual destructor.
- virtual `AAX_IPParameterValue * CloneValue () const =0`
Clone the parameter's value to a new `AAX_IPParameterValue` object.

Identification methods

- virtual `AAX_CParamID Identifier () const =0`
Returns the parameter's unique identifier.
- virtual void `SetName (const AAX_CString &name)=0`
Sets the parameter's display name.
- virtual const `AAX_CString & Name () const =0`
Returns the parameter's display name.
- virtual void `AddShortenedName (const AAX_CString &name)=0`

- virtual const [AAX_CString](#) & [ShortenedName](#) (int32_t iNumCharacters) const =0
Returns the parameter's shortened display name.
- virtual void [ClearShortenedNames](#) ()=0
Clears the internal list of shortened display names.

Automation methods

- virtual bool [Automatable](#) () const =0
Returns true if the parameter is automatable, false if it is not.
- virtual void [SetAutomationDelegate](#) ([AAX_IAutomationDelegate](#) *iAutomationDelegate)=0
Sets the automation delegate (if one is required)
- virtual void [Touch](#) ()=0
Signals the automation system that a control has been touched.
- virtual void [Release](#) ()=0
Signals the automation system that a control has been released.

Taper methods

- virtual void [SetNormalizedValue](#) (double newNormalizedValue)=0
Sets a parameter value using it's normalized representation.
- virtual double [GetNormalizedValue](#) () const =0
Returns the normalized representation of the parameter's current real value.
- virtual void [SetNormalizedDefaultValue](#) (double normalizedDefault)=0
Sets the parameter's default value using its normalized representation.
- virtual double [GetNormalizedDefaultValue](#) () const =0
Returns the normalized representation of the parameter's real default value.
- virtual void [SetToDefaultValue](#) ()=0
Restores the state of this parameter to its default value.
- virtual void [SetNumberOfSteps](#) (uint32_t numSteps)=0
Sets the number of discrete steps for this parameter.
- virtual uint32_t [GetNumberOfSteps](#) () const =0
Returns the number of discrete steps used by the parameter.
- virtual uint32_t [GetStepValue](#) () const =0
Returns the current step for the current value of the parameter.
- virtual double [GetNormalizedValueFromStep](#) (uint32_t iStep) const =0
Returns the normalized value for a given step.
- virtual uint32_t [GetStepValueFromNormalizedValue](#) (double normalizedValue) const =0
Returns the step value for a normalized value of the parameter.
- virtual void [SetStepValue](#) (uint32_t iStep)=0
Returns the current step for the current value of the parameter.

Display methods

This functionality is most often used by GUIs, but can also be useful for state serialization.

- virtual bool [GetValueString](#) ([AAX_CString](#) *valueString) const =0
Serializes the parameter value into a string.
- virtual bool [GetValueString](#) (int32_t iMaxNumChars, [AAX_CString](#) *valueString) const =0
Serializes the parameter value into a string, size hint included.
- virtual bool [GetNormalizedValueFromBool](#) (bool value, double *normalizedValue) const =0
Converts a bool to a normalized parameter value.
- virtual bool [GetNormalizedValueFromInt32](#) (int32_t value, double *normalizedValue) const =0
Converts an integer to a normalized parameter value.
- virtual bool [GetNormalizedValueFromFloat](#) (float value, double *normalizedValue) const =0
Converts a float to a normalized parameter value.
- virtual bool [GetNormalizedValueFromDouble](#) (double value, double *normalizedValue) const =0
Converts a double to a normalized parameter value.

- virtual bool [GetNormalizedValueFromString](#) (const [AAX_CString](#) &valueString, double *normalizedValue) const =0
Converts a given string to a normalized parameter value.
- virtual bool [GetBoolFromNormalizedValue](#) (double normalizedValue, bool *value) const =0
Converts a normalized parameter value to a bool representing the corresponding real value.
- virtual bool [GetInt32FromNormalizedValue](#) (double normalizedValue, int32_t *value) const =0
Converts a normalized parameter value to an integer representing the corresponding real value.
- virtual bool [GetFloatFromNormalizedValue](#) (double normalizedValue, float *value) const =0
Converts a normalized parameter value to a float representing the corresponding real value.
- virtual bool [GetDoubleFromNormalizedValue](#) (double normalizedValue, double *value) const =0
Converts a normalized parameter value to a double representing the corresponding real value.
- virtual bool [GetStringFromNormalizedValue](#) (double normalizedValue, [AAX_CString](#) &valueString) const =0
Converts a normalized parameter value to a string representing the corresponding real value.
- virtual bool [GetStringFromNormalizedValue](#) (double normalizedValue, int32_t iMaxNumChars, [AAX_CString](#) &valueString) const =0
Converts a normalized parameter value to a string representing the corresponding real, size hint included. value.
- virtual bool [SetValueFromString](#) (const [AAX_CString](#) &newValueString)=0
Converts a string to a real parameter value and sets the parameter to this value.

Typed accessors

- virtual bool [GetValueAsBool](#) (bool *value) const =0
Retrieves the parameter's value as a bool.
- virtual bool [GetValueAsInt32](#) (int32_t *value) const =0
Retrieves the parameter's value as an int32_t.
- virtual bool [GetValueAsFloat](#) (float *value) const =0
Retrieves the parameter's value as a float.
- virtual bool [GetValueAsDouble](#) (double *value) const =0
Retrieves the parameter's value as a double.
- virtual bool [GetValueAsString](#) ([AAX_IString](#) *value) const =0
Retrieves the parameter's value as a string.
- virtual bool [SetValueWithBool](#) (bool value)=0
Sets the parameter's value as a bool.
- virtual bool [SetValueWithInt32](#) (int32_t value)=0
Sets the parameter's value as an int32_t.
- virtual bool [SetValueWithFloat](#) (float value)=0
Sets the parameter's value as a float.
- virtual bool [SetValueWithDouble](#) (double value)=0
Sets the parameter's value as a double.
- virtual bool [SetValueWithString](#) (const [AAX_IString](#) &value)=0
Sets the parameter's value as a string.
- virtual void [SetType](#) ([AAX_EParameterType](#) iControlType)=0
Sets the type of this parameter.
- virtual [AAX_EParameterType](#) [GetType](#) () const =0
Returns the type of this parameter as an AAX_EParameterType.
- virtual void [SetOrientation](#) ([AAX_EParameterOrientation](#) iOrientation)=0
Sets the orientation of this parameter.
- virtual [AAX_EParameterOrientation](#) [GetOrientation](#) () const =0
Returns the orientation of this parameter.
- virtual void [SetTaperDelegate](#) ([AAX_ITaperDelegateBase](#) &inTaperDelegate, bool inPreserveValue)=0
Sets the parameter's taper delegate.
- virtual void [SetDisplayDelegate](#) ([AAX_IDisplayDelegateBase](#) &inDisplayDelegate)=0
Sets the parameter's display delegate.

Host interface methods

- virtual void [UpdateNormalizedValue](#) (double newNormalizedValue)=0
Sets the parameter's state given a normalized value.

14.120.2 Constructor & Destructor Documentation

14.120.2.1 ~AAX_IParameter()

```
virtual AAX_IParameter::~~AAX_IParameter ( ) [inline], [virtual]
```

Virtual destructor.

Note

This destructor MUST be virtual to prevent memory leaks.

14.120.3 Member Function Documentation

14.120.3.1 CloneValue()

```
virtual AAX_IParameterValue * AAX_IParameter::CloneValue ( ) const [pure virtual]
```

Clone the parameter's value to a new [AAX_IParameterValue](#) object.

The returned object is independent from the [AAX_IParameter](#). For example, changing the state of the returned object will not result in a change to the original [AAX_IParameter](#).

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

Referenced by [AAX_CMonolithicParameters::GenerateCoefficients\(\)](#).

Here is the caller graph for this function:



14.120.3.2 Identifier()

```
virtual AAX_CParamID AAX_IParameter::Identifier ( ) const [pure virtual]
```

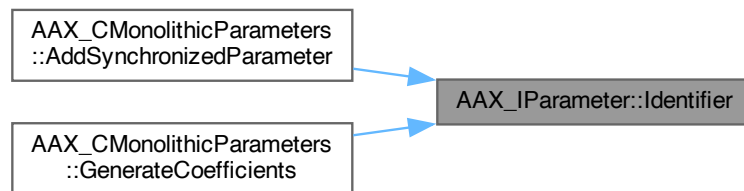
Returns the parameter's unique identifier.

This unique ID is used by the [Parameter Manager](#) and by outside applications to uniquely identify and target control messages. This value may not be changed after the parameter has been constructed.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

Referenced by [AAX_CMonolithicParameters::AddSynchronizedParameter\(\)](#), and [AAX_CMonolithicParameters::GenerateCoefficients](#).

Here is the caller graph for this function:



14.120.3.3 SetName()

```
virtual void AAX_IParameter::SetName (
    const AAX_CString & name ) [pure virtual]
```

Sets the parameter's display name.

This name is used for display only, it is not used for indexing or identifying the parameter. This name may be changed after the parameter has been created, but display name changes may not be recognized by all AAX hosts.

Parameters

in	<i>name</i>	Display name that will be assigned to the parameter
----	-------------	---

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.4 Name()

```
virtual const AAX_CString & AAX_IParameter::Name ( ) const [pure virtual]
```

Returns the parameter's display name.

Note

This method returns a const reference in order to prevent a string copy. Do not cast away the const to change this value.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.5 AddShortenedName()

```
virtual void AAX_IParameter::AddShortenedName (
    const AAX_CString & name ) [pure virtual]
```

Sets the parameter's shortened display name.

This name is used for display only, it is not used for indexing or identifying the parameter. These names show up when the host asks for shorter length parameter names for display on Control Surfaces or other string length constrained situations.

Parameters

in	<i>name</i>	Shortened display names that will be assigned to the parameter
----	-------------	--

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.6 ShortenedName()

```
virtual const AAX_CString & AAX_IParameter::ShortenedName (
    int32_t iNumCharacters ) const [pure virtual]
```

Returns the parameter's shortened display name.

Note

This method returns a const reference in order to prevent a string copy. Do not cast away the const to change this value.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.7 ClearShortenedNames()

```
virtual void AAX_IParameter::ClearShortenedNames ( ) [pure virtual]
```

Clears the internal list of shortened display names.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.8 Automatable()

```
virtual bool AAX_IParacter::Automatable ( ) const [pure virtual]
```

Returns true if the parameter is automatable, false if it is not.

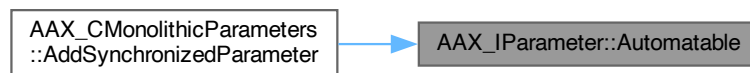
Note

Subclasses that return true in this method must support host-based automation.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

Referenced by [AAX_CMonolithicParameters::AddSynchronizedParameter\(\)](#).

Here is the caller graph for this function:

**14.120.3.9 SetAutomationDelegate()**

```
virtual void AAX_IParacter::SetAutomationDelegate (
    AAX_IAutomationDelegate * iAutomationDelegate ) [pure virtual]
```

Sets the automation delegate (if one is required)

Parameters

in	<i>iAutomationDelegate</i>	A reference to the parameter manager's automation delegate interface
----	----------------------------	--

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.10 Touch()

```
virtual void AAX_IParacter::Touch ( ) [pure virtual]
```

Signals the automation system that a control has been touched.

Call this method in response to GUI events that begin editing, such as a mouse down. After this method has been called you are free to call [SetNormalizedValue\(\)](#) as much as you need, e.g. in order to respond to subsequent mouse moved events. Call [Release\(\)](#) to free the parameter for updates from other controls.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.11 Release()

```
virtual void AAX_IParameter::Release ( ) [pure virtual]
```

Signals the automation system that a control has been released.

Call this method in response to GUI events that complete editing, such as a mouse up. Once this method has been called you should not call [SetNormalizedValue\(\)](#) again until after the next call to [Touch\(\)](#).

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.12 SetNormalizedValue()

```
virtual void AAX_IParameter::SetNormalizedValue (
    double newNormalizedValue ) [pure virtual]
```

Sets a parameter value using it's normalized representation.

For more information regarding normalized values, see [Parameter Manager](#)

Parameters

in	<i>newNormalizedValue</i>	New value (normalized) to which the parameter will be set
----	---------------------------	---

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.13 GetNormalizedValue()

```
virtual double AAX_IParameter::GetNormalizedValue ( ) const [pure virtual]
```

Returns the normalized representation of the parameter's current real value.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.14 SetNormalizedDefaultValue()

```
virtual void AAX_IParameter::SetNormalizedDefaultValue (
    double normalizedDefault ) [pure virtual]
```

Sets the parameter's default value using its normalized representation.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.15 GetNormalizedDefaultValue()

```
virtual double AAX_IPParameter::GetNormalizedDefaultValue ( ) const [pure virtual]
```

Returns the normalized representation of the parameter's real default value.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.16 SetToDefaultValue()

```
virtual void AAX_IPParameter::SetToDefaultValue ( ) [pure virtual]
```

Restores the state of this parameter to its default value.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.17 SetNumberOfSteps()

```
virtual void AAX_IPParameter::SetNumberOfSteps (
    uint32_t numSteps ) [pure virtual]
```

Sets the number of discrete steps for this parameter.

Stepped parameter values are useful for discrete parameters and for "jumping" events such as mouse wheels, page up/down, etc. The parameter's step size is used to specify the coarseness of those changes.

Note

numSteps MUST be greater than zero. All other values may be considered an error by the host.

Parameters

in	<i>numSteps</i>	The number of steps that the parameter will use
----	-----------------	---

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.18 GetNumberOfSteps()

```
virtual uint32_t AAX_IPParameter::GetNumberOfSteps ( ) const [pure virtual]
```

Returns the number of discrete steps used by the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.19 GetStepValue()

```
virtual uint32_t AAX_IParameter::GetStepValue ( ) const [pure virtual]
```

Returns the current step for the current value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.20 GetNormalizedValueFromStep()

```
virtual double AAX_IParameter::GetNormalizedValueFromStep (
    uint32_t iStep ) const [pure virtual]
```

Returns the normalized value for a given step.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.21 GetStepValueFromNormalizedValue()

```
virtual uint32_t AAX_IParameter::GetStepValueFromNormalizedValue (
    double normalizedValue ) const [pure virtual]
```

Returns the step value for a normalized value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.22 SetStepValue()

```
virtual void AAX_IParameter::SetStepValue (
    uint32_t iStep ) [pure virtual]
```

Returns the current step for the current value of the parameter.

See [SetNumberOfSteps\(\)](#) for more information about parameter steps.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.23 GetValueString() [1/2]

```
virtual bool AAX_IParameter::GetValueString (
    AAX_CString * valueString ) const [pure virtual]
```

Serializes the parameter value into a string.

Parameters

out	<i>valueString</i>	A string representing the parameter's real value
-----	--------------------	--

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.24 GetValueString() [2/2]

```
virtual bool AAX_IParacter::GetValueString (
    int32_t iMaxNumChars,
    AAX_CString * valueString ) const [pure virtual]
```

Serializes the parameter value into a string, size hint included.

Parameters

in	<i>iMaxNumChars</i>	A size hint for the size of the string being requested. Useful for control surfaces and other limited area text fields. (make sure that size of desired string also has room for null termination)
out	<i>valueString</i>	A string representing the parameter's real value

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.25 GetNormalizedValueFromBool()

```
virtual bool AAX_IParacter::GetNormalizedValueFromBool (
    bool value,
    double * normalizedValue ) const [pure virtual]
```

Converts a bool to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.26 GetNormalizedValueFromInt32()

```
virtual bool AAX_IParameter::GetNormalizedValueFromInt32 (
    int32_t value,
    double * normalizedValue ) const [pure virtual]
```

Converts an integer to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.27 GetNormalizedValueFromFloat()

```
virtual bool AAX_IParameter::GetNormalizedValueFromFloat (
    float value,
    double * normalizedValue ) const [pure virtual]
```

Converts a float to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.28 GetNormalizedValueFromDouble()

```
virtual bool AAX_IParacter::GetNormalizedValueFromDouble (
    double value,
    double * normalizedValue ) const [pure virtual]
```

Converts a double to a normalized parameter value.

Parameters

in	<i>value</i>	A value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with value

Return values

<i>true</i>	The value conversion was successful
<i>false</i>	The value conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.29 GetNormalizedValueFromString()

```
virtual bool AAX_IParacter::GetNormalizedValueFromString (
    const AAX\_CString & valueString,
    double * normalizedValue ) const [pure virtual]
```

Converts a given string to a normalized parameter value.

Parameters

in	<i>valueString</i>	A string representing a possible real value for the parameter
out	<i>normalizedValue</i>	The normalized parameter value associated with valueString

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CStatelessParameter](#), and [AAX_CParameter< T >](#).

14.120.3.30 GetBoolFromNormalizedValue()

```
virtual bool AAX_IParameter::GetBoolFromNormalizedValue (
    double normalizedValue,
    bool * value ) const [pure virtual]
```

Converts a normalized parameter value to a bool representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.31 GetInt32FromNormalizedValue()

```
virtual bool AAX_IParameter::GetInt32FromNormalizedValue (
    double normalizedValue,
    int32_t * value ) const [pure virtual]
```

Converts a normalized parameter value to an integer representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to int32_t was successful
<i>false</i>	The conversion to int32_t was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.32 GetFloatFromNormalizedValue()

```
virtual bool AAX_IParameter::GetFloatFromNormalizedValue (
    double normalizedValue,
    float * value ) const [pure virtual]
```

Converts a normalized parameter value to a float representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.33 GetDoubleFromNormalizedValue()

```
virtual bool AAX_IParameter::GetDoubleFromNormalizedValue (
    double normalizedValue,
    double * value ) const [pure virtual]
```

Converts a normalized parameter value to a double representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	The normalized value to convert
out	<i>value</i>	The converted value. Set only if conversion is successful.

Return values

<i>true</i>	The conversion to double was successful
<i>false</i>	The conversion to double was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.34 GetStringFromNormalizedValue() [1/2]

```
virtual bool AAX_IParameter::GetStringFromNormalizedValue (
    double normalizedValue,
    AAX_CString & valueString ) const [pure virtual]
```

Converts a normalized parameter value to a string representing the corresponding real value.

Parameters

in	<i>normalizedValue</i>	A normalized parameter value
out	<i>valueString</i>	A string representing the parameter value associated with <i>normalizedValue</i>

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.35 GetStringFromNormalizedValue() [2/2]

```
virtual bool AAX_IParameter::GetStringFromNormalizedValue (
    double normalizedValue,
    int32_t iMaxNumChars,
    AAX_CString & valueString ) const [pure virtual]
```

Converts a normalized parameter value to a string representing the corresponding real, size hint included. *value*.

Parameters

in	<i>normalizedValue</i>	A normalized parameter value
in	<i>iMaxNumChars</i>	A size hint for the size of the string being requested. Useful for control surfaces and other limited area text fields. (make sure that size of desired string also has room for null termination)
out	<i>valueString</i>	A string representing the parameter value associated with <i>normalizedValue</i>

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.36 SetValueFromString()

```
virtual bool AAX_IParameter::SetValueFromString (
    const AAX_CString & newValueString ) [pure virtual]
```

Converts a string to a real parameter value and sets the parameter to this value.

Parameters

in	<i>newValueString</i>	A string representing the parameter's new real value
----	-----------------------	--

Return values

<i>true</i>	The string conversion was successful
<i>false</i>	The string conversion was unsuccessful

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.37 GetValueAsBool()

```
virtual bool AAX_IParacter::GetValueAsBool (
    bool * value ) const [pure virtual]
```

Retrieves the parameter's value as a bool.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to bool was successful
<i>false</i>	The conversion to bool was unsuccessful

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.38 GetValueAsInt32()

```
virtual bool AAX_IParacter::GetValueAsInt32 (
    int32_t * value ) const [pure virtual]
```

Retrieves the parameter's value as an int32_t.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to int32_t was successful
<i>false</i>	The conversion to int32_t was unsuccessful

Implemented in [AAX_CStatelessParameter](#), and [AAX_CParameter< T >](#).

14.120.3.39 GetValueAsFloat()

```
virtual bool AAX_IParameter::GetValueAsFloat (
    float * value ) const [pure virtual]
```

Retrieves the parameter's value as a float.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to float was successful
false	The conversion to float was unsuccessful

Implemented in [AAX_CStatelessParameter](#), and [AAX_CParameter< T >](#).

14.120.3.40 GetValueAsDouble()

```
virtual bool AAX_IParameter::GetValueAsDouble (
    double * value ) const [pure virtual]
```

Retrieves the parameter's value as a double.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to double was successful
false	The conversion to double was unsuccessful

Implemented in [AAX_CStatelessParameter](#), and [AAX_CParameter< T >](#).

14.120.3.41 GetValueAsString()

```
virtual bool AAX_IParameter::GetValueAsString (
    AAX_IString * value ) const [pure virtual]
```

Retrieves the parameter's value as a string.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to string was successful
false	The conversion to string was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CStatelessParameter](#), and [AAX_CParameter< T >](#).

14.120.3.42 SetValueWithBool()

```
virtual bool AAX_IParameter::SetValueWithBool (  
    bool value ) [pure virtual]
```

Sets the parameter's value as a bool.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion from bool was successful
false	The conversion from bool was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.43 SetValueWithInt32()

```
virtual bool AAX_IParameter::SetValueWithInt32 (  
    int32_t value ) [pure virtual]
```

Sets the parameter's value as an int32_t.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion from int32_t was successful
false	The conversion from int32_t was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.44 SetValueWithFloat()

```
virtual bool AAX_IParameter::SetValueWithFloat (
    float value ) [pure virtual]
```

Sets the parameter's value as a float.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from float was successful
<i>false</i>	The conversion from float was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.45 SetValueWithDouble()

```
virtual bool AAX_IParameter::SetValueWithDouble (
    double value ) [pure virtual]
```

Sets the parameter's value as a double.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from double was successful
<i>false</i>	The conversion from double was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.46 SetValueWithString()

```
virtual bool AAX_IParameter::SetValueWithString (
    const AAX\_IString & value ) [pure virtual]
```

Sets the parameter's value as a string.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion from string was successful
<i>false</i>	The conversion from string was unsuccessful

Implemented in [AAX_CParameter< T >](#), [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.47 SetType()

```
virtual void AAX_IParacter::SetType (
    AAX_EParameterType iControlType ) [pure virtual]
```

Sets the type of this parameter.

See [GetType](#) for use cases

Parameters

in	<i>iControlType</i>	The parameter's new type as an AAX_EParameterType
----	---------------------	---

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.48 GetType()

```
virtual AAX_EParameterType AAX_IParacter::GetType ( ) const [pure virtual]
```

Returns the type of this parameter as an AAX_EParameterType.

Todo Document use cases for control type

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.49 SetOrientation()

```
virtual void AAX_IParacter::SetOrientation (
    AAX_EParameterOrientation iOrientation ) [pure virtual]
```

Sets the orientation of this parameter.

Parameters

in	<i>iOrientation</i>	The parameter's new orientation
----	---------------------	---------------------------------

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.50 GetOrientation()

```
virtual AAX\_EParameterOrientation AAX_IParameter::GetOrientation ( ) const [pure virtual]
```

Returns the orientation of this parameter.

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

14.120.3.51 SetTaperDelegate()

```
virtual void AAX_IParameter::SetTaperDelegate (
    AAX\_ITaperDelegateBase & inTaperDelegate,
    bool inPreserveValue ) [pure virtual]
```

Sets the parameter's taper delegate.

Parameters

in	<i>inTaperDelegate</i>	A reference to the parameter's new taper delegate
in	<i>inPreserveValue</i>	

Todo Document this parameter

Implemented in [AAX_CStatelessParameter](#), and [AAX_CParameter< T >](#).

14.120.3.52 SetDisplayDelegate()

```
virtual void AAX_IParameter::SetDisplayDelegate (
    AAX\_IDisplayDelegateBase & inDisplayDelegate ) [pure virtual]
```

Sets the parameter's display delegate.

Parameters

in	<i>inDisplayDelegate</i>	A reference to the parameter's new display delegate
----	--------------------------	---

Implemented in [AAX_CStatelessParameter](#), and [AAX_CParameter< T >](#).

14.120.3.53 UpdateNormalizedValue()

```
virtual void AAX_IParameter::UpdateNormalizedValue (
    double newNormalizedValue ) [pure virtual]
```

Sets the parameter's state given a normalized value.

This is the second half of the parameter setting operation that is initiated with a call to SetValue(). Parameters should not be set directly using this method; instead, use SetValue().

Parameters

in	<i>newNormalizedValue</i>	Normalized value that will be used to set the parameter's new state
----	---------------------------	---

Implemented in [AAX_CParameter< T >](#), and [AAX_CStatelessParameter](#).

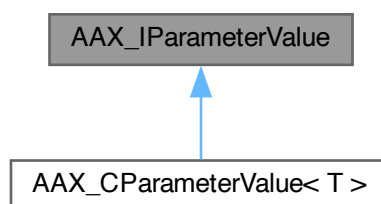
The documentation for this class was generated from the following file:

- [AAX_IParameter.h](#)

14.121 AAX_IParameterValue Class Reference

```
#include <AAX_IParameter.h>
```

Inheritance diagram for AAX_IParameterValue:



14.121.1 Description

An abstract interface representing a parameter value of arbitrary type.

:Internal to the AAX SDK

See also

[AAX_IParameter](#)

Public Member Functions

- virtual [~AAX_IParаметerValue](#) ()
Virtual destructor.
- virtual [AAX_IParаметerValue](#) * [Clone](#) () const =0
Clones the parameter object.
- virtual [AAX_CParamID Identifier](#) () const =0
Returns the parameter's unique identifier.

Typed accessors

- virtual bool [GetValueAsBool](#) (bool *value) const =0
Retrieves the parameter's value as a bool.
- virtual bool [GetValueAsInt32](#) (int32_t *value) const =0
Retrieves the parameter's value as an int32_t.
- virtual bool [GetValueAsFloat](#) (float *value) const =0
Retrieves the parameter's value as a float.
- virtual bool [GetValueAsDouble](#) (double *value) const =0
Retrieves the parameter's value as a double.
- virtual bool [GetValueAsString](#) ([AAX_IString](#) *value) const =0
Retrieves the parameter's value as a string.

14.121.2 Constructor & Destructor Documentation

14.121.2.1 ~AAX_IParаметerValue()

```
virtual AAX_IParаметerValue::~~AAX_IParаметerValue ( ) [inline], [virtual]
```

Virtual destructor.

Note

This destructor MUST be virtual to prevent memory leaks.

14.121.3 Member Function Documentation

14.121.3.1 Clone()

```
virtual AAX_IParаметerValue * AAX_IParаметerValue::Clone ( ) const [pure virtual]
```

Clones the parameter object.

Note

Does NOT set the automation delegate on the clone; ownership of the automation delegate and parameter registration/unregistration stays with the original parameter

Implemented in [AAX_CParameterValue< T >](#).

14.121.3.2 Identifier()

```
virtual AAX_CParamID AAX_IParameterValue::Identifier ( ) const [pure virtual]
```

Returns the parameter's unique identifier.

This unique ID is used by the [Parameter Manager](#) and by outside applications to uniquely identify and target control messages. This value may not be changed after the parameter has been constructed.

Implemented in [AAX_CParameterValue< T >](#).

14.121.3.3 GetValueAsBool()

```
virtual bool AAX_IParameterValue::GetValueAsBool (
    bool * value ) const [pure virtual]
```

Retrieves the parameter's value as a bool.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to bool was successful
false	The conversion to bool was unsuccessful

Implemented in [AAX_CParameterValue< T >](#), and [AAX_CParameterValue< T >](#).

14.121.3.4 GetValueAsInt32()

```
virtual bool AAX_IParameterValue::GetValueAsInt32 (
    int32_t * value ) const [pure virtual]
```

Retrieves the parameter's value as an int32_t.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to int32_t was successful
false	The conversion to int32_t was unsuccessful

Implemented in [AAX_CParameterValue< T >](#), and [AAX_CParameterValue< T >](#).

14.121.3.5 GetValueAsFloat()

```
virtual bool AAX_IParаметerValue::GetValueAsFloat (
    float * value ) const [pure virtual]
```

Retrieves the parameter's value as a float.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to float was successful
<i>false</i>	The conversion to float was unsuccessful

Implemented in [AAX_CParameterValue< T >](#), and [AAX_CParameterValue< T >](#).

14.121.3.6 GetValueAsDouble()

```
virtual bool AAX_IParаметerValue::GetValueAsDouble (
    double * value ) const [pure virtual]
```

Retrieves the parameter's value as a double.

Parameters

out	<i>value</i>	The parameter's real value. Set only if conversion is successful.
-----	--------------	---

Return values

<i>true</i>	The conversion to double was successful
<i>false</i>	The conversion to double was unsuccessful

Implemented in [AAX_CParameterValue< T >](#), and [AAX_CParameterValue< T >](#).

14.121.3.7 GetValueAsString()

```
virtual bool AAX_IParаметerValue::GetValueAsString (
    AAX\_IString * value ) const [pure virtual]
```

Retrieves the parameter's value as a string.

Parameters

out	value	The parameter's real value. Set only if conversion is successful.
-----	-------	---

Return values

true	The conversion to string was successful
false	The conversion to string was unsuccessful

Implemented in [AAX_CParameterValue< T >](#), and [AAX_CParameterValue< T >](#).

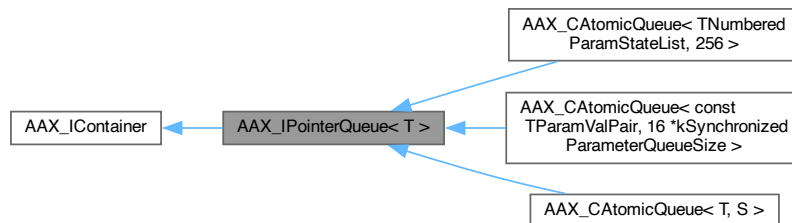
The documentation for this class was generated from the following file:

- [AAX_IParameter.h](#)

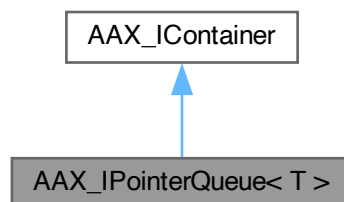
14.122 AAX_IPointerQueue< T > Class Template Reference

```
#include <AAX_IPointerQueue.h>
```

Inheritance diagram for AAX_IPointerQueue< T >:



Collaboration diagram for AAX_IPointerQueue< T >:



14.122.1 Description

```
template<typename T>  
class AAX_IPointerQueue< T >
```

Abstract interface for a basic FIFO queue of pointers-to-objects

Public Types

- typedef T [template_type](#)
The type used for this template instance.
- typedef T * [value_type](#)
The type of values stored in this queue.

Public Types inherited from [AAX_IContainer](#)

- enum [EStatus](#) {
 [eStatus_Success](#) = 0 ,
 [eStatus_Overflow](#) = 1 ,
 [eStatus_NotInitialized](#) = 2 ,
 [eStatus_Unavailable](#) = 3 ,
 [eStatus_Unsupported](#) = 4 }

Public Member Functions

- virtual [~AAX_IPointerQueue](#) ()
- virtual void [Clear](#) ()=0
- virtual [AAX_IContainer::EStatus Push](#) ([value_type](#) inElem)=0
- virtual [value_type Pop](#) ()=0
- virtual [value_type Peek](#) () const =0

Public Member Functions inherited from [AAX_IContainer](#)

- virtual [~AAX_IContainer](#) ()
- virtual void [Clear](#) ()=0

14.122.2 Member Typedef Documentation

14.122.2.1 [template_type](#)

```
template<typename T >  
typedef T AAX\_IPointerQueue< T >::template_type
```

The type used for this template instance.

14.122.2.2 value_type

```
template<typename T >
typedef T* AAX_IPointerQueue< T >::value_type
```

The type of values stored in this queue.

14.122.3 Constructor & Destructor Documentation

14.122.3.1 ~AAX_IPointerQueue()

```
template<typename T >
virtual AAX_IPointerQueue< T >::~~AAX_IPointerQueue ( ) [inline], [virtual]
```

14.122.4 Member Function Documentation

14.122.4.1 Clear()

```
template<typename T >
virtual void AAX_IPointerQueue< T >::Clear ( ) [pure virtual]
```

Note

This operation is NOT atomic

This does NOT call the destructor for any pointed-to elements; it only clears the pointer values in the queue

Implements [AAX_IContainer](#).

Implemented in [AAX_CAtomicQueue< T, S >](#), [AAX_CAtomicQueue< TNumberedParamStateList, 256 >](#), and [AAX_CAtomicQueue< const TParamValPair, 16 *kSynchronizedParameterQueueSize >](#).

14.122.4.2 Push()

```
template<typename T >
virtual AAX_IContainer::EStatus AAX_IPointerQueue< T >::Push (
    value_type inElem ) [pure virtual]
```

Push an element onto the queue

Call from: Write thread

Returns

[AAX_IContainer::EStatus_Success](#) if the push succeeded

Implemented in [AAX_CAtomicQueue< T, S >](#).

14.122.4.3 Pop()

```
template<typename T >
virtual value_type AAX_IPointerQueue< T >::Pop ( ) [pure virtual]
```

Pop the front element from the queue

Call from: Read thread

Returns

NULL if no element is available

Implemented in [AAX_CAtomicQueue< T, S >](#), [AAX_CAtomicQueue< TNumberedParamStateList, 256 >](#), and [AAX_CAtomicQueue< const TParamValPair, 16 *kSynchronizedParameterQueueSize >](#).

14.122.4.4 Peek()

```
template<typename T >
virtual value_type AAX_IPointerQueue< T >::Peek ( ) const [pure virtual]
```

Get the current top element without popping it off of the queue

Call from: Read thread

Note

This value will change if another thread calls [Pop\(\)](#)

Implemented in [AAX_CAtomicQueue< T, S >](#), [AAX_CAtomicQueue< TNumberedParamStateList, 256 >](#), and [AAX_CAtomicQueue< const TParamValPair, 16 *kSynchronizedParameterQueueSize >](#).

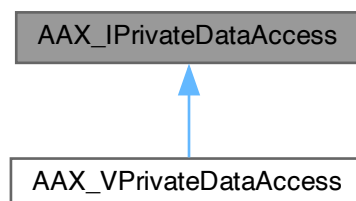
The documentation for this class was generated from the following file:

- [AAX_IPointerQueue.h](#)

14.123 AAX_IPrivateDataAccess Class Reference

```
#include <AAX_IPrivateDataAccess.h>
```

Inheritance diagram for AAX_IPrivateDataAccess:



14.123.1 Description

Interface to data access provided by host to plug-in.

:Implemented by the AAX Host

WARNING: [AAX_IPrivateDataAccess](#) objects are not reference counted and are not guaranteed to exist beyond the scope of the method(s) they are passed into.

See also

[AAX_IACFEffEffectDirectData::TimerWakeup](#)

Public Member Functions

- virtual [~AAX_IPrivateDataAccess](#) ()
- virtual [AAX_Result ReadPortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, void *outBuffer)=0
Read data directly from DSP at the given port.
- virtual [AAX_Result WritePortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, const void *inBuffer)=0
Write data directly to DSP at the given port.

14.123.2 Constructor & Destructor Documentation

14.123.2.1 ~AAX_IPrivateDataAccess()

```
virtual AAX_IPrivateDataAccess::~AAX_IPrivateDataAccess ( ) [inline], [virtual]
```

14.123.3 Member Function Documentation

14.123.3.1 ReadPortDirect()

```
virtual AAX_Result AAX_IPrivateDataAccess::ReadPortDirect (
    AAX_CFieldIndex inFieldIndex,
    const uint32_t inOffset,
    const uint32_t inSize,
    void * outBuffer ) [pure virtual]
```

Read data directly from DSP at the given port.

Note

Blocking

Parameters

in	<i>inFieldIndex</i>	The port to read from.
in	<i>inOffset</i>	Offset into data to start reading.
in	<i>inSize</i>	Amount of data to read (in bytes).
out	<i>outBuffer</i>	Pointer to storage for data to be read into.

Implemented in [AAX_VPrivateDataAccess](#).

14.123.3.2 WritePortDirect()

```
virtual AAX_Result AAX_IPrivateDataAccess::WritePortDirect (
    AAX_CFieldIndex inFieldIndex,
    const uint32_t inOffset,
    const uint32_t inSize,
    const void * inBuffer ) [pure virtual]
```

Write data directly to DSP at the given port.

Note

Blocking

Parameters

in	<i>inFieldIndex</i>	The port to write to.
in	<i>inOffset</i>	Offset into data to begin writing.
in	<i>inSize</i>	Amount of data to write (in bytes).
in	<i>inBuffer</i>	Pointer to data being written.

Implemented in [AAX_VPrivateDataAccess](#).

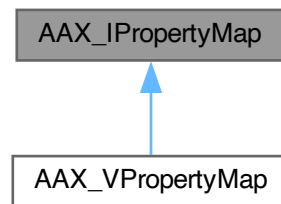
The documentation for this class was generated from the following file:

- [AAX_IPrivateDataAccess.h](#)

14.124 AAX_IPropertyMap Class Reference

```
#include <AAX_IPropertyMap.h>
```


Inheritance diagram for AAX_IPropertyMap:



14.124.1 Description

Generic plug-in description property map.

:Implemented by the AAX Host

Property Maps are used to associate specific sets of properties with plug-in description interfaces. For example, an audio processing component might register mono and stereo callbacks, or Native and TI callbacks, assigning each `ProcessProc` the applicable property mapping. This allows the host to determine the correct callback to use depending on the environment in which the plug-in is instantiated.

AAX does not require that every value in AAX IPropertyMap be assigned by the developer. Unassigned properties do not have defined default values; if a specific value is not assigned to one of an element's properties then the element must support any value for that property. For example, if an audio processing component does not define its callback's audio buffer length property, the host will assume that the callback will support any buffer length.

- To create a new property map: [AAX_IComponentDescriptor::NewPropertyMap\(\)](#)
- To copy an existing property map: [AAX_IComponentDescriptor::DuplicatePropertyMap\(\)](#)

Public Member Functions

- virtual [~AAX_IPropertyMap](#) ()
- virtual [AAX_CBoolean GetProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) *outValue) const =0
Get a property value from a property map.
- virtual [AAX_CBoolean GetPointerProperty](#) ([AAX_EProperty](#) inProperty, const void **outValue) const =0
Get a property value from a property map with a pointer-sized value.
- virtual [AAX_Result AddProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) inValue)=0
Add a property to a property map.
- virtual [AAX_Result AddPointerProperty](#) ([AAX_EProperty](#) inProperty, const void *inValue)=0
Add a property to a property map with a pointer-sized value.
- virtual [AAX_Result AddPointerProperty](#) ([AAX_EProperty](#) inProperty, const char *inValue)=0
Add a property to a property map with a pointer-sized value.
- virtual [AAX_Result RemoveProperty](#) ([AAX_EProperty](#) inProperty)=0
Remove a property from a property map.

- virtual [AAX_Result](#) [AddPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPluginIdentifierTriad](#) *inPluginIDs, uint32_t inNumPluginIDs)=0
Add an array of plug-in IDs to a property map.
- virtual [AAX_CBoolean](#) [GetPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPluginIdentifierTriad](#) **outPluginIDs, uint32_t *outNumPluginIDs) const =0
Get an array of plug-in IDs from a property map.
- virtual [IACFUnknown](#) * [GetIUnknown](#) ()=0

14.124.2 Constructor & Destructor Documentation

14.124.2.1 ~AAX_IPropertyMap()

```
virtual AAX_IPropertyMap::~AAX_IPropertyMap ( ) [inline], [virtual]
```

14.124.3 Member Function Documentation

14.124.3.1 GetProperty()

```
virtual AAX\_CBoolean AAX_IPropertyMap::GetProperty (
    AAX\_EProperty inProperty,
    AAX\_CPropertyValue * outValue ) const [pure virtual]
```

Get a property value from a property map.

Returns true if the selected property is supported, false if it is not

Parameters

in	<i>inProperty</i>	The property ID
out	<i>outValue</i>	The property value

Implemented in [AAX_VPropertyMap](#).

14.124.3.2 GetPointerProperty()

```
virtual AAX\_CBoolean AAX_IPropertyMap::GetPointerProperty (
    AAX\_EProperty inProperty,
    const void ** outValue ) const [pure virtual]
```

Get a property value from a property map with a pointer-sized value.

Returns true if the selected property is supported, false if it is not

Parameters

in	<i>inProperty</i>	The property ID
out	<i>outValue</i>	The property value

Implemented in [AAX_VPropertyMap](#).

14.124.3.3 AddProperty()

```
virtual AAX_Result AAX_IPropertyMap::AddProperty (
    AAX_EProperty inProperty,
    AAX_CPropertyValue inValue ) [pure virtual]
```

Add a property to a property map.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

Implemented in [AAX_VPropertyMap](#).

Referenced by [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

Here is the caller graph for this function:

**14.124.3.4 AddPointerProperty()** [1/2]

```
virtual AAX_Result AAX_IPropertyMap::AddPointerProperty (
    AAX_EProperty inProperty,
    const void * inValue ) [pure virtual]
```

Add a property to a property map with a pointer-sized value.

Use this method to add properties which require a pointer-sized value. Do not use this method to add a property unless a pointer-sized value is explicitly specified in the property documentation.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

Implemented in [AAX_VPropertyMap](#).

14.124.3.5 AddPointerProperty() [2/2]

```
virtual AAX_Result AAX_IPropertyMap::AddPointerProperty (
    AAX_EProperty inProperty,
    const char * inValue ) [pure virtual]
```

Add a property to a property map with a pointer-sized value.

Use this method to add properties which require a pointer-sized value. Do not use this method to add a property unless a pointer-sized value is explicitly specified in the property documentation.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

Implemented in [AAX_VPropertyMap](#).

14.124.3.6 RemoveProperty()

```
virtual AAX_Result AAX_IPropertyMap::RemoveProperty (
    AAX_EProperty inProperty ) [pure virtual]
```

Remove a property from a property map.

Parameters

in	<i>inProperty</i>	The property ID.
----	-------------------	------------------

Implemented in [AAX_VPropertyMap](#).

14.124.3.7 AddPropertyWithIDArray()

```
virtual AAX_Result AAX_IPropertyMap::AddPropertyWithIDArray (
    AAX_EProperty inProperty,
    const AAX_SPlugInIdentifierTriad * inPluginIDs,
    uint32_t inNumPluginIDs ) [pure virtual]
```

Add an array of plug-in IDs to a property map.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inPluginIDs</i>	An array of AAX_SPlugInIdentifierTriad
in	<i>inNumPluginIDs</i>	The length of iPluginIDs

Implemented in [AAX_VPropertyMap](#).

14.124.3.8 GetPropertyWithIDArray()

```
virtual AAX_CBoolean AAX_IPropertyMap::GetPropertyWithIDArray (
    AAX_EProperty inProperty,
    const AAX_SPlugInIdentifierTriad ** outPluginIDs,
    uint32_t * outNumPluginIDs ) const [pure virtual]
```

Get an array of plug-in IDs from a property map.

Parameters

in	<i>inProperty</i>	The property ID.
out	<i>outPluginIDs</i>	A pointer that will be set to reference an array of AAX_SPlugInIdentifierTriad
in	<i>outNumPluginIDs</i>	The length of oPluginIDs

Implemented in [AAX_VPropertyMap](#).

14.124.3.9 GetIUnknown()

```
virtual IACFUnknown * AAX_IPropertyMap::GetIUnknown ( ) [pure virtual]
```

Returns the most up-to-date underlying interface

Implemented in [AAX_VPropertyMap](#).

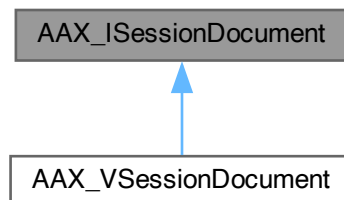
The documentation for this class was generated from the following file:

- [AAX_IPropertyMap.h](#)

14.125 AAX_ISessionDocument Class Reference

```
#include <AAX_ISessionDocument.h>
```

Inheritance diagram for AAX_ISessionDocument:



14.125.1 Description

Interface representing information in a host session document.

This interface wraps the versioned interfaces defined in [AAX_IACFSessionDocument.h](#) and provides additional convenience functions providing session data back in the expected format.

See also

[AAX_ISessionDocumentClient](#)

Classes

- class [TempoMap](#)

Public Member Functions

- virtual [~AAX_ISessionDocument](#) ()=default
- virtual bool [Valid](#) () const =0
Check whether this session document is valid.
- virtual std::unique_ptr< [TempoMap](#) const > [GetTempoMap](#) ()=0
Get a copy of the document's tempo map.
- virtual [AAX_Result](#) [GetDocumentData](#) ([AAX_DocumentData_UID](#) const &inDataType, [IACFUnknown](#) **outData)=0

14.125.2 Constructor & Destructor Documentation

14.125.2.1 ~AAX_I_SessionDocument()

```
virtual AAX_I_SessionDocument::~~AAX_I_SessionDocument ( ) [virtual], [default]
```

14.125.3 Member Function Documentation

14.125.3.1 Valid()

```
virtual bool AAX_I_SessionDocument::Valid ( ) const [pure virtual]
```

Check whether this session document is valid.

Implemented in [AAX_V_SessionDocument](#).

14.125.3.2 GetTempoMap()

```
virtual std::unique_ptr< TempoMap const > AAX_I_SessionDocument::GetTempoMap ( ) [pure virtual]
```

Get a copy of the document's tempo map.

Returns

A [TempoMap](#) interface representing a copy of the current tempo map.

`nullptr` if the host does not support tempo map data or if an error occurred.

Implemented in [AAX_V_SessionDocument](#).

14.125.3.3 GetDocumentData()

```
virtual AAX\_Result AAX_I_SessionDocument::GetDocumentData (
    AAX\_DocumentData\_UID const & inDataType,
    IACFUnknown ** outData ) [pure virtual]
```

Get document data of a generic type

Similar to `QueryInterface()` but uses a data type identifier rather than a true IID

The provided interface has already had a reference added, so be careful not to add an additional reference:

```
ACFPtr<MyType> ptr;
IACFUnknown * docDataPtr(nullptr);
if (AAX_SUCCESS == doc->GetDocumentData(dataUID, &docDataPtr) && docDataPtr) {
    ptr.attach(std::static_cast<MyType*>(docDataPtr)); // attach does not AddRef
}
```

Parameters

in	<i>inDataType</i>	The type of the document data requested
out	<i>outData</i>	An interface providing the requested data, or <code>nullptr</code> if the host does not support or cannot provide the requested data type. The reference count has been incremented on this object on behalf of the caller, so the caller must not add an additional reference count and must decrement the reference count on this object to release it. For information about which interface to expect for each requested data type, see the documentation for that data type.

Implemented in [AAX_VSessionDocument](#).

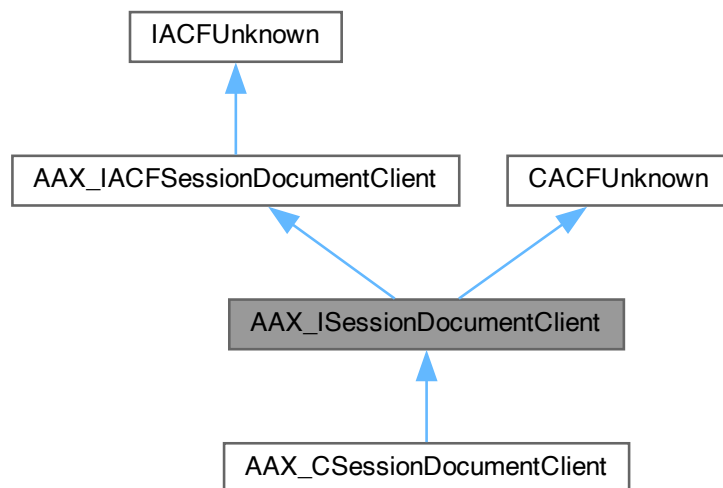
The documentation for this class was generated from the following file:

- [AAX_ISessionDocument.h](#)

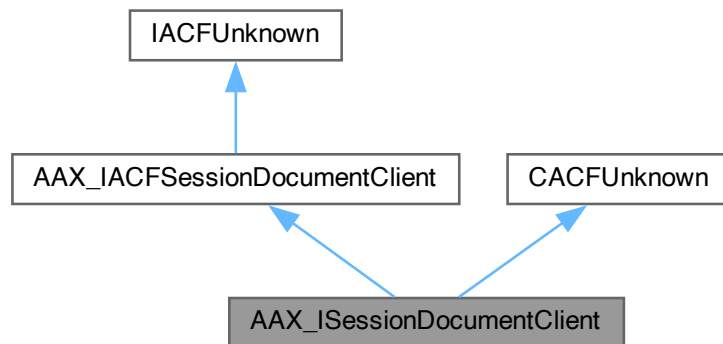
14.126 AAX_ISessionDocumentClient Class Reference

```
#include <AAX_ISessionDocumentClient.h>
```

Inheritance diagram for AAX_ISessionDocumentClient:



Collaboration diagram for AAX_I_SessionDocumentClient:



14.126.1 Description

Interface representing a client of the session document interface.

For example, a plug-in implementation that makes calls on the session document interface provided by the host.

Public Member Functions

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acfIID](#) &riid
- [AAX_DELETE](#) ([AAX_I_SessionDocumentClient](#) &operator=(const [AAX_I_SessionDocumentClient](#) &))

Public Member Functions inherited from [AAX_IACFSessionDocumentClient](#)

- virtual [AAX_Result Initialize](#) ([IACFUnknown](#) *iUnknown)=0
- virtual [AAX_Result Uninitialize](#) (void)=0
- virtual [AAX_Result SetSessionDocument](#) ([IACFUnknown](#) *iSessionDocument)=0
Sets or removes a session document.
- virtual [AAX_Result NotificationReceived](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, uint32_t inNotificationDataSize)=0
Notification Hook.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Public Attributes

- void **ppvObjOut [override](#)

14.126.2 Member Function Documentation

14.126.2.1 ACF_DECLARE_STANDARD_UNKNOWN()

```
AAX_I_SessionDocumentClient::ACF_DECLARE_STANDARD_UNKNOWN ( ) const &
```

14.126.2.2 AAX_DELETE()

```
AAX_I_SessionDocumentClient::AAX_DELETE (
    AAX_I_SessionDocumentClient & operator = (const AAX_I_SessionDocumentClient &) )
```

14.126.3 Member Data Documentation

14.126.3.1 override

```
void** ppvObjOut AAX_I_SessionDocumentClient::override
```

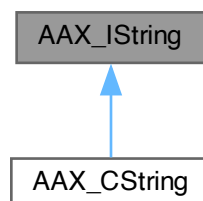
The documentation for this class was generated from the following file:

- [AAX_I_SessionDocumentClient.h](#)

14.127 AAX_IString Class Reference

```
#include <AAX_IString.h>
```

Inheritance diagram for AAX_IString:



14.127.1 Description

A simple string container that can be passed across a binary boundary. This class, for simplicity, is not versioned and thus can never change.

For a real string implementation, see [AAX_CString](#), which inherits from this interface, but provides a much richer string interface.

This object is not versioned with ACF for a variety of reasons, but the biggest implication of that is that THIS INTERFACE CAN NEVER CHANGE!

Public Member Functions

- virtual [~AAX_IString](#) ()
- virtual uint32_t [Length](#) () const =0
- virtual uint32_t [MaxLength](#) () const =0
- virtual const char * [Get](#) () const =0
- virtual void [Set](#) (const char *iString)=0
- virtual [AAX_IString](#) & [operator=](#) (const [AAX_IString](#) &iOther)=0
- virtual [AAX_IString](#) & [operator=](#) (const char *iString)=0

14.127.2 Constructor & Destructor Documentation

14.127.2.1 ~AAX_IString()

```
virtual AAX_IString::~~AAX_IString ( ) [inline], [virtual]
```

Virtual Destructor

14.127.3 Member Function Documentation

14.127.3.1 Length()

```
virtual uint32_t AAX_IString::Length ( ) const [pure virtual]
```

Length methods

Implemented in [AAX_CString](#).

Referenced by [AAX::String2Binary\(\)](#).

Here is the caller graph for this function:



14.127.3.2 MaxLength()

```
virtual uint32_t AAX_IString::MaxLength ( ) const [pure virtual]
```

Implemented in [AAX_CString](#).

14.127.3.3 Get()

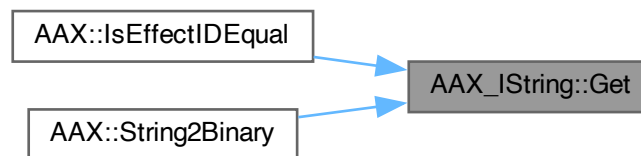
```
virtual const char * AAX_IString::Get ( ) const [pure virtual]
```

C string methods

Implemented in [AAX_CString](#).

Referenced by [AAX::IsEffectIDEqual\(\)](#), and [AAX::String2Binary\(\)](#).

Here is the caller graph for this function:



14.127.3.4 Set()

```
virtual void AAX_IString::Set (
    const char * iString ) [pure virtual]
```

Implemented in [AAX_CString](#).

14.127.3.5 operator=() [1/2]

```
virtual AAX\_IString & AAX_IString::operator= (
    const AAX\_IString & iOther ) [pure virtual]
```

Assignment operators

Implemented in [AAX_CString](#).

14.127.3.6 operator=() [2/2]

```
virtual AAX_IString & AAX_IString::operator= (
    const char * iString ) [pure virtual]
```

Implemented in [AAX_CString](#).

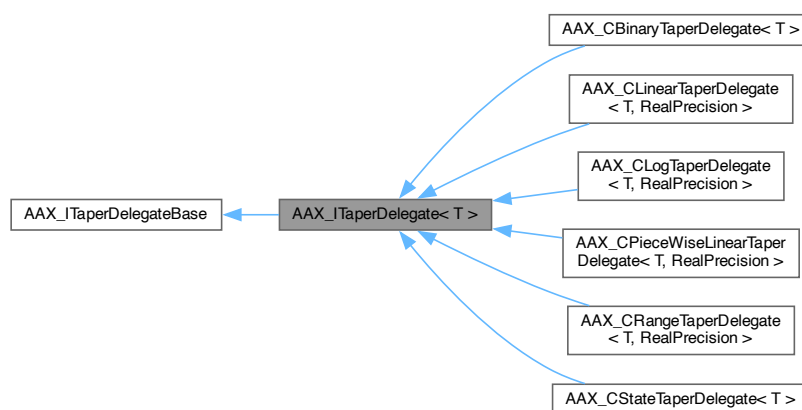
The documentation for this class was generated from the following file:

- [AAX_IString.h](#)

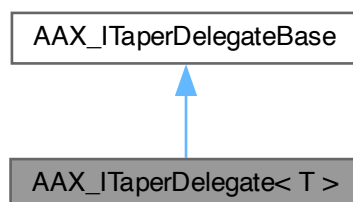
14.128 AAX_ITaperDelegate< T > Class Template Reference

```
#include <AAX_ITaperDelegate.h>
```

Inheritance diagram for AAX_ITaperDelegate< T >:



Collaboration diagram for AAX_ITaperDelegate< T >:



14.128.1 Description

```
template<typename T>
class AAX_ITaperDelegate< T >
```

Taper delegate interface template

Classes for conversion to and from normalized parameter values.

Taper delegates are used to convert real parameter values to and from their normalized representations. All taper delegates implement the `AAX_ITaperDelegate<T>` interface template, which contains two conversion functions:

```
virtual T      NormalizedToReal(double normalizedValue) const = 0;
virtual double RealToNormalized(T realValue) const = 0;
```

In addition, tapers may incorporate logical value constraints via the following interface methods:

```
virtual T      GetMaximumValue() const = 0;
virtual T      GetMinimumValue() const = 0;
virtual T      ConstrainRealValue(T value) const = 0;
```

For more information, see the [AAX_ITaperDelegate](#) class documentation.

Public Member Functions

- virtual [AAX_ITaperDelegate * Clone](#) () const =0
Constructs and returns a copy of the taper delegate.
- virtual T [GetMaximumValue](#) () const =0
Returns the taper's maximum real value.
- virtual T [GetMinimumValue](#) () const =0
Returns the taper's minimum real value.
- virtual T [ConstrainRealValue](#) (T value) const =0
Applies a constraint to the value and returns the constrained value.
- virtual T [NormalizedToReal](#) (double normalizedValue) const =0
Converts a normalized value to a real value.
- virtual double [RealToNormalized](#) (T realValue) const =0
Normalizes a real parameter value.

Public Member Functions inherited from [AAX_ITaperDelegateBase](#)

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

14.128.2 Member Function Documentation

14.128.2.1 Clone()

```
template<typename T >
virtual AAX_ITaperDelegate * AAX_ITaperDelegate< T >::Clone ( ) const [pure virtual]
```

Constructs and returns a copy of the taper delegate.

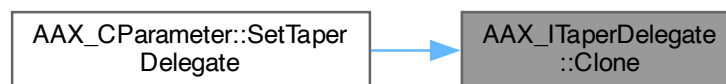
In general, this method's implementation can use a simple copy constructor:

```
template <typename T>
AAX_CSubclassTaperDelegate<T>* AAX_CSubclassTaperDelegate<T>::Clone() const
{
    return new AAX_CSubclassTaperDelegate(*this);
}
```

Implemented in [AAX_CBinaryTaperDelegate< T >](#), [AAX_CLinearTaperDelegate< T, RealPrecision >](#), [AAX_CLogTaperDelegate< T, RealPrecision >](#), [AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >](#), [AAX_CRangeTaperDelegate< T, RealPrecision >](#), and [AAX_CStateTaperDelegate< T >](#).

Referenced by [AAX_CParameter< T >::SetTaperDelegate\(\)](#).

Here is the caller graph for this function:



14.128.2.2 GetMaximumValue()

```
template<typename T >
virtual T AAX_ITaperDelegate< T >::GetMaximumValue ( ) const [pure virtual]
```

Returns the taper's maximum real value.

Implemented in [AAX_CBinaryTaperDelegate< T >](#), [AAX_CLinearTaperDelegate< T, RealPrecision >](#), [AAX_CLogTaperDelegate< T, RealPrecision >](#), [AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >](#), [AAX_CRangeTaperDelegate< T, RealPrecision >](#), and [AAX_CStateTaperDelegate< T >](#).

14.128.2.3 GetMinimumValue()

```
template<typename T >
virtual T AAX_ITaperDelegate< T >::GetMinimumValue ( ) const [pure virtual]
```

Returns the taper's minimum real value.

Implemented in [AAX_CBinaryTaperDelegate< T >](#), [AAX_CLinearTaperDelegate< T, RealPrecision >](#), [AAX_CLogTaperDelegate< T, RealPrecision >](#), [AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >](#), [AAX_CRangeTaperDelegate< T, RealPrecision >](#), and [AAX_CStateTaperDelegate< T >](#).

14.128.2.4 ConstrainRealValue()

```
template<typename T >
virtual T AAX\_ITaperDelegate< T >::ConstrainRealValue (
    T value ) const [pure virtual]
```

Applies a constraint to the value and returns the constrained value.

This method is useful if the taper requires a constraint beyond simple minimum and maximum real value limits.

Note

This is the function that should actually enforces the constraints in [NormalizeToReal\(\)](#) and [RealToNormalized\(\)](#).

Parameters

in	<i>value</i>	The unconstrained value
----	--------------	-------------------------

Implemented in [AAX_CBinaryTaperDelegate< T >](#), [AAX_CLinearTaperDelegate< T, RealPrecision >](#), [AAX_CLogTaperDelegate< T, RealPrecision >](#), [AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >](#), [AAX_CRangeTaperDelegate< T, RealPrecision >](#), and [AAX_CStateTaperDelegate< T >](#).

14.128.2.5 NormalizedToReal()

```
template<typename T >
virtual T AAX\_ITaperDelegate< T >::NormalizedToReal (
    double normalizedValue ) const [pure virtual]
```

Converts a normalized value to a real value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [RealToNormalized\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>normalizedValue</i>	The normalized value that will be converted
----	------------------------	---

Implemented in [AAX_CBinaryTaperDelegate< T >](#), [AAX_CLinearTaperDelegate< T, RealPrecision >](#), [AAX_CLogTaperDelegate< T, RealPrecision >](#), [AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >](#), [AAX_CRangeTaperDelegate< T, RealPrecision >](#), and [AAX_CStateTaperDelegate< T >](#).

14.128.2.6 RealToNormalized()

```
template<typename T >
virtual double AAX\_ITaperDelegate< T >::RealToNormalized (
    T realValue ) const [pure virtual]
```


Normalizes a real parameter value.

This is where the actual taper algorithm is implemented.

This function should perform the exact inverse of [NormalizedToReal\(\)](#), to within the roundoff precision of the individual taper implementation.

Parameters

in	<i>realValue</i>	The real parameter value that will be normalized
----	------------------	--

Implemented in [AAX_CBinaryTaperDelegate< T >](#), [AAX_CLinearTaperDelegate< T, RealPrecision >](#), [AAX_CLogTaperDelegate< T, RealPrecision >](#), [AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >](#), [AAX_CRangeTaperDelegate< T, RealPrecision >](#), and [AAX_CStateTaperDelegate< T >](#).

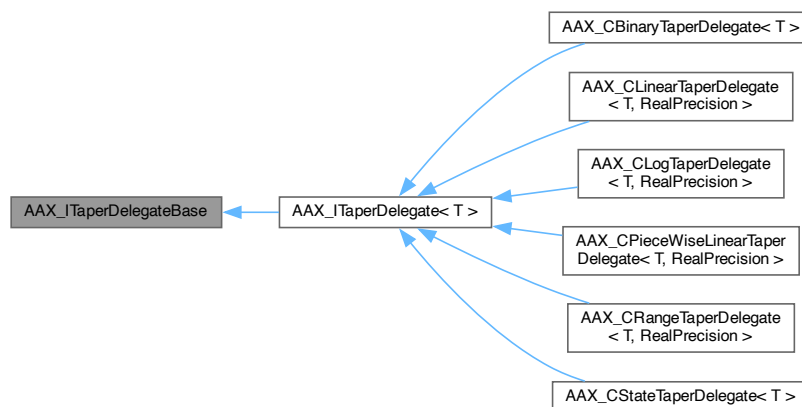
The documentation for this class was generated from the following file:

- [AAX_ITaperDelegate.h](#)

14.129 AAX_ITaperDelegateBase Class Reference

```
#include <AAX_ITaperDelegate.h>
```

Inheritance diagram for AAX_ITaperDelegateBase:



14.129.1 Description

Defines the taper conversion behavior for a parameter.

:Internal to the AAX SDK

This interface represents a delegate class to be used in conjunction with [AAX_IParameter](#). [AAX_IParameter](#) delegates all conversion operations between normalized and real parameter values to classes that meet this interface. You can think of [AAX_ITaperDelegate](#) subclasses as simple taper conversion routines that enable a specific taper or range conversion function on an arbitrary parameter.

To demonstrate the use of this interface, we will examine a simple call routine into a parameter:

1. The host application calls into the plug-in's [AAX_CParameterManager](#) with a Parameter ID and a new normalized parameter value. This new value could be coming from an automation lane, a control surface, or any other parameter control; from the plug-in's perspective, these are all identical.
2. The [AAX_CParameterManager](#) finds the specified [AAX_CParameter](#) and calls [AAX_IParameter::SetNormalizedValue\(\)](#) on that parameter
3. [AAX_IParameter::SetNormalizedValue\(\)](#) results in a call into the parameter's concrete taper delegate to convert the normalized value to a real value.

Using this pattern, the parameter manager is able to use real parameter values without actually knowing how to perform the conversion between normalized and real values.

The inverse of the above example can also happen, e.g. when a control is updated from within the data model. In this case, the parameter can call into its concrete taper delegate in order to normalize the updated value, which can then be passed on to any observers that require normalized values, such as the host app.

For more information about the parameter manager, see the [Parameter Manager](#) documentation page.

Public Member Functions

- virtual [~AAX_ITaperDelegateBase](#) ()
Virtual destructor.

14.129.2 Constructor & Destructor Documentation

14.129.2.1 ~AAX_ITaperDelegateBase()

```
virtual AAX_ITaperDelegateBase::~~AAX_ITaperDelegateBase ( ) [inline], [virtual]
```

Virtual destructor.

Note

This destructor MUST be virtual to prevent memory leaks.

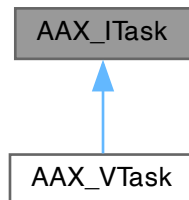
The documentation for this class was generated from the following file:

- [AAX_ITaperDelegate.h](#)

14.130 AAX_ITask Class Reference

```
#include <AAX_ITask.h>
```

Inheritance diagram for AAX_ITask:



14.130.1 Description

Interface representing a request to perform a task.

:Implemented by the AAX Host

Used by the [task agent](#).

This interface describes a task request and provides a way for the agent to express one or more results of the task as well as the progress of the task.

This interface is open-ended for both inputs and outputs. The host and agent must use common definitions for specific task types, their possible arguments, and the expected results.

Public Member Functions

- virtual [~AAX_ITask](#) ()=default
- virtual [AAX_Result](#) [GetType](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_IACFDataBuffer](#) const * [GetArgumentOfType](#) ([AAX_CTypeID](#) iType) const =0
- virtual [AAX_Result](#) [SetProgress](#) (float iProgress)=0
- virtual float [GetProgress](#) () const =0
- virtual [AAX_Result](#) [AddResult](#) ([AAX_IACFDataBuffer](#) const *iResult)=0
 - Attach result data to this task.*
- virtual [AAX_ITask](#) * [SetDone](#) ([AAX_TaskCompletionStatus](#) iStatus)=0
 - Inform the host that the task is completed.*

14.130.2 Constructor & Destructor Documentation

14.130.2.1 ~AAX_ITask()

```
virtual AAX_ITask::~AAX_ITask ( ) [virtual], [default]
```

14.130.3 Member Function Documentation

14.130.3.1 GetType()

```
virtual AAX_Result AAX_ITask::GetType (
    AAX_CTypeID * oType ) const [pure virtual]
```

An identifier defining the type of the requested task

Parameters

out	<i>oType</i>	The type of this task request
-----	--------------	-------------------------------

Implemented in [AAX_VTask](#).

14.130.3.2 GetArgumentOfType()

```
virtual AAX_IACFDataBuffer const * AAX_ITask::GetArgumentOfType (
    AAX_CTypeID iType ) const [pure virtual]
```

Additional information defining the request, depending on the task type

Parameters

in	<i>iType</i>	The type of argument requested. Possible argument types, if any, and the resulting data buffer format must be defined per task type.
----	--------------	--

Returns

The requested argument data, or nullptr. This data buffer's type ID is expected to match *iType* . The caller takes ownership of this object.

Implemented in [AAX_VTask](#).

14.130.3.3 SetProgress()

```
virtual AAX_Result AAX_ITask::SetProgress (
    float iProgress ) [pure virtual]
```

Inform the host about the current status of the task

Parameters

in	<i>iProgress</i>	A value between 0 (no progress) and 1 (complete)
----	------------------	--

Implemented in [AAX_VTask](#).

14.130.3.4 GetProgress()

```
virtual float AAX_ITask::GetProgress ( ) const [pure virtual]
```

Returns the current progress

Implemented in [AAX_VTask](#).

14.130.3.5 AddResult()

```
virtual AAX_Result AAX_ITask::AddResult (
    AAX_IACFDataBuffer const * iResult ) [pure virtual]
```

Attach result data to this task.

This can be called multiple times to add multiple types of results to a single task.

The host may process the result data immediately or may wait for the task to complete.

The plug-in is expected to release the data buffer upon making this call. At a minimum, the data buffer must not be changed after this call is made. See `ACFPtr::inArg()`

Parameters

in	<i>iResult</i>	A buffer containing the result data. Expected result types, if any, and their data buffer format must be defined per task type.
----	----------------	---

Implemented in [AAX_VTask](#).

14.130.3.6 SetDone()

```
virtual AAX_ITask * AAX_ITask::SetDone (
    AAX_TaskCompletionStatus iStatus ) [pure virtual]
```

Inform the host that the task is completed.

If successful, returns a null pointer. Otherwise, returns a pointer back to the same object. This is the expected usage pattern:

```
// release the task on success, retain it on failure
myTask = myTask->SetDone(status);
```

Parameters

in	<i>iStatus</i>	The final status of the task. This indicates to the host whether or not the task was performed as requested.
----	----------------	--

Implemented in [AAX_VTask](#).

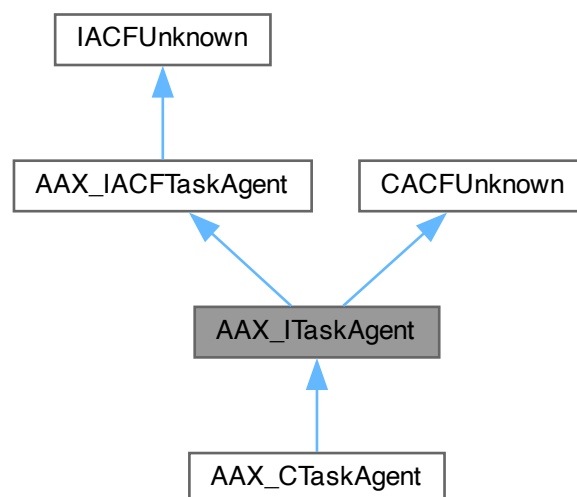
The documentation for this class was generated from the following file:

- [AAX_ITask.h](#)

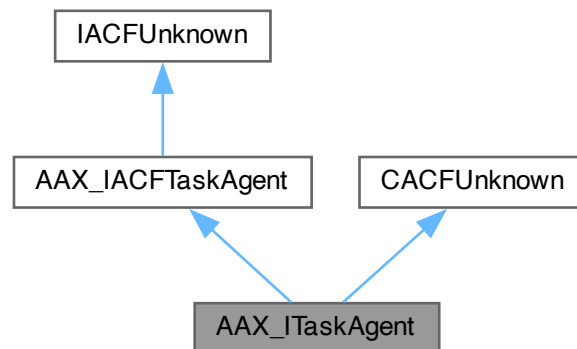
14.131 AAX_ITaskAgent Class Reference

```
#include <AAX_ITaskAgent.h>
```

Inheritance diagram for AAX_ITaskAgent:



Collaboration diagram for AAX_ITaskAgent:



14.131.1 Description

Interface for a component that accepts task requests.

:Implemented by the Plug-In

The task agent is expected to complete the requested tasks asynchronously and to provide progress and completion details via calls on the [AAX_IACFTask](#) interface as the tasks proceed.

See also

[AAX_ITask](#)

Public Member Functions

- [ACF_DECLARE_STANDARD_UNKNOWN](#) () ACFMETHOD(InternalQueryInterface)(const [acflID](#) &riid
- [AAX_DELETE](#) (AAX_ITaskAgent &operator=(const [AAX_ITaskAgent](#) &))

Public Member Functions inherited from [AAX_IACFTaskAgent](#)

- virtual [AAX_Result Initialize](#) (IACFUnknown *iController)=0
- virtual [AAX_Result Uninitialize](#) ()=0
- virtual [AAX_Result AddTask](#) (IACFUnknown *iTask)=0
- virtual [AAX_Result CancelAllTasks](#) ()=0

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

Public Attributes

- void **ppvObjOut [AAX_OVERRIDE](#)

14.131.2 Member Function Documentation

14.131.2.1 ACF_DECLARE_STANDARD_UNKNOWN()

```
AAX_ITaskAgent::ACF_DECLARE_STANDARD_UNKNOWN ( ) const &
```

14.131.2.2 AAX_DELETE()

```
AAX_ITaskAgent::AAX_DELETE (
    AAX\_ITaskAgent & operator = (const AAX\_ITaskAgent &) )
```

14.131.3 Member Data Documentation

14.131.3.1 AAX_OVERRIDE

```
void** ppvObjOut AAX_ITaskAgent::AAX_OVERRIDE
```

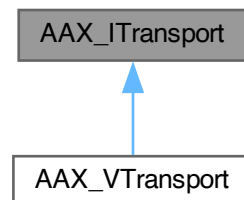
The documentation for this class was generated from the following file:

- [AAX_ITaskAgent.h](#)

14.132 AAX_ITransport Class Reference

```
#include <AAX_ITransport.h>
```

Inheritance diagram for AAX_ITransport:



14.132.1 Description

Interface to information about the host's transport state.

:Implemented by the AAX Host

Plug-ins that use this interface should describe [AAX_eProperty_UsesTransport](#) as 1

Classes that inherit from [AAX_CEffectParameters](#) or [AAX_CEffectGUI](#) can use [AAX_CEffectParameters::Transport\(\)](#) / [AAX_CEffectGUI::Transport\(\)](#) to access this interface. This interface is used as a local interface to the [AAX_VTransport](#) versioned implementation, which dispatches calls to the appropriate host-supplied versioned transport interface depending on which features are supported by the host. See [AAX_CEffectParameters::Initialize\(\)](#) for an example.

A copy of this interface may also be obtained directly from the host using [AAX_IMIDINode::GetTransport\(\)](#). However, in this case the interface is not versioned, so the host and the plugin may not agree on the interface. This can lead to undefined behavior. See the documentation at [AAX_IMIDINode::GetTransport\(\)](#) for more information.

Public Member Functions

- virtual [~AAX_ITransport](#) ()
Virtual destructor.
- virtual [AAX_Result GetCurrentTempo](#) (double *TempoBPM) const =0
CALL: Gets the current tempo.
- virtual [AAX_Result GetCurrentMeter](#) (int32_t *MeterNumerator, int32_t *MeterDenominator) const =0
CALL: Gets the current meter.
- virtual [AAX_Result IsTransportPlaying](#) (bool *isPlaying) const =0
CALL: Indicates whether or not the transport is playing back.
- virtual [AAX_Result GetCurrentTickPosition](#) (int64_t *TickPosition) const =0
CALL: Gets the current tick position.

- virtual [AAX_Result GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0
CALL: Gets current information on loop playback.
- virtual [AAX_Result GetCurrentNativeSampleLocation](#) (int64_t *SampleLocation) const =0
CALL: Gets the current playback location of the native audio engine.
- virtual [AAX_Result GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding tick position.
- virtual [AAX_Result GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t SampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding bar and beat position.
- virtual [AAX_Result GetTicksPerQuarter](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per quarter note.
- virtual [AAX_Result GetCurrentTicksPerBeat](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per beat.
- virtual [AAX_Result GetTimelineSelectionStartPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the beginning of the current transport selection.
- virtual [AAX_Result GetTimeCodeInfo](#) (AAX_EFrameRate *oFrameRate, int32_t *oOffset) const =0
CALL: Retrieves the current time code frame rate and offset.
- virtual [AAX_Result GetFeetFramesInfo](#) (AAX_EFeetFramesRate *oFeetFramesRate, int64_t *oOffset) const =0
CALL: Retrieves the current timecode feet/frames rate and offset.
- virtual [AAX_Result IsMetronomeEnabled](#) (int32_t *isEnabled) const =0
Sets isEnabled to true if the metronome is enabled.
- virtual [AAX_Result GetHDTimeCodeInfo](#) (AAX_EFrameRate *oHDFrameRate, int64_t *oHDOffset) const =0
CALL: Retrieves the current HD time code frame rate and offset.
- virtual [AAX_Result RequestTransportStart](#) ()=0
CALL: Request that the host transport start playback.
- virtual [AAX_Result RequestTransportStop](#) ()=0
CALL: Request that the host transport stop playback.
- virtual [AAX_Result GetTimelineSelectionEndPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the end of the current transport selection.

14.132.2 Constructor & Destructor Documentation

14.132.2.1 ~AAX_ITransport()

```
virtual AAX_ITransport::~~AAX_ITransport ( ) [inline], [virtual]
```

Virtual destructor.

Note

This destructor MUST be virtual to prevent memory leaks.

14.132.3 Member Function Documentation

14.132.3.1 GetCurrentTempo()

```
virtual AAX_Result AAX_ITransport::GetCurrentTempo (
    double * TempoBPM ) const [pure virtual]
```

CALL: Gets the current tempo.

Returns the tempo corresponding to the current position of the transport counter

Note

The resolution of the tempo returned here is based on the host's tempo resolution, so it will match the tempo displayed in the host. Use [GetCurrentTicksPerBeat\(\)](#) to calculate the tempo resolution note.

Parameters

out	<i>TempoBPM</i>	The current tempo in beats per minute
-----	-----------------	---------------------------------------

Implemented in [AAX_VTransport](#).

14.132.3.2 GetCurrentMeter()

```
virtual AAX_Result AAX_ITransport::GetCurrentMeter (
    int32_t * MeterNumerator,
    int32_t * MeterDenominator ) const [pure virtual]
```

CALL: Gets the current meter.

Returns the meter corresponding to the current position of the transport counter

Parameters

out	<i>MeterNumerator</i>	The numerator portion of the meter
out	<i>MeterDenominator</i>	The denominator portion of the meter

Implemented in [AAX_VTransport](#).

14.132.3.3 IsTransportPlaying()

```
virtual AAX_Result AAX_ITransport::IsTransportPlaying (
    bool * isPlaying ) const [pure virtual]
```

CALL: Indicates whether or not the transport is playing back.

Parameters

out	<i>isPlaying</i>	true if the transport is currently in playback
-----	------------------	--

Implemented in [AAX_VTransport](#).

14.132.3.4 GetCurrentTickPosition()

```
virtual AAX_Result AAX_ITransport::GetCurrentTickPosition (
    int64_t * TickPosition ) const [pure virtual]
```

CALL: Gets the current tick position.

Returns the current tick position corresponding to the current transport position. One "Tick" is represented here as 1/960000 of a quarter note. That is, there are 960,000 of these ticks in a quarter note.

Host Compatibility Notes The tick resolution here is different than that of the tick displayed in Pro Tools. "Display ticks" (as they are called) are 1/960 of a quarter note.

Parameters

out	<i>TickPosition</i>	The tick position value
-----	---------------------	-------------------------

Implemented in [AAX_VTransport](#).

14.132.3.5 GetCurrentLoopPosition()

```
virtual AAX_Result AAX_ITransport::GetCurrentLoopPosition (
    bool * bLooping,
    int64_t * LoopStartTick,
    int64_t * LoopEndTick ) const [pure virtual]
```

CALL: Gets current information on loop playback.

Host Compatibility Notes This does not indicate anything about the status of the "Loop Record" option. Even when the host is configured to loop playback, looping may not occur if certain conditions are not met (i.e. the length of the selection is too short)

Parameters

out	<i>bLooping</i>	true if the host is configured to loop playback
out	<i>LoopStartTick</i>	The starting tick position of the selection being looped (see GetCurrentTickPosition())
out	<i>LoopEndTick</i>	The ending tick position of the selection being looped (see GetCurrentTickPosition())

Implemented in [AAX_VTransport](#).

14.132.3.6 GetCurrentNativeSampleLocation()

```
virtual AAX_Result AAX_ITransport::GetCurrentNativeSampleLocation (
    int64_t * SampleLocation ) const [pure virtual]
```

CALL: Gets the current playback location of the native audio engine.

When called from a ProcessProc render callback, this method will provide the absolute sample location at the beginning of the callback's audio buffers.

When called from [AAX_IEffectParameters::RenderAudio_Hybrid\(\)](#), this method will provide the absolute sample location for the samples in the method's **output** audio buffers. To calculate the absolute sample location for the samples in the method's input buffers (i.e. the timeline location where the samples originated) subtract the value provided by [AAX_IController::GetHybridSignalLatency\(\)](#) from this value.

When called from a non-real-time thread, this method will provide the current location of the samples being processed by the plug-in's ProcessProc on its real-time processing thread.

Note

This method only returns a value during playback. It cannot be used to determine, e.g., the location of the timeline selector while the host is not in playback.

Parameters

out	<i>SampleLocation</i>	Absolute sample location of the first sample in the current native processing buffer
-----	-----------------------	--

Implemented in [AAX_VTransport](#).

14.132.3.7 GetCustomTickPosition()

```
virtual AAX_Result AAX_ITransport::GetCustomTickPosition (
    int64_t * oTickPosition,
    int64_t iSampleLocation ) const [pure virtual]
```

CALL: Given an absolute sample position, gets the corresponding tick position.

Host Compatibility Notes There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Parameters

out	<i>oTickPosition</i>	the timeline tick position corresponding to <i>iSampleLocation</i>
in	<i>iSampleLocation</i>	An absolute sample location (see GetCurrentNativeSampleLocation())

Implemented in [AAX_VTransport](#).

14.132.3.8 GetBarBeatPosition()

```
virtual AAX_Result AAX_ITransport::GetBarBeatPosition (
    int32_t * Bars,
    int32_t * Beats,
    int64_t * DisplayTicks,
    int64_t SampleLocation ) const [pure virtual]
```

CALL: Given an absolute sample position, gets the corresponding bar and beat position.

Host Compatibility Notes There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Parameters

out	<i>Bars</i>	The bar corresponding to <i>SampleLocation</i>
out	<i>Beats</i>	The beat corresponding to <i>SampleLocation</i>
out	<i>DisplayTicks</i>	The ticks corresponding to <i>SampleLocation</i>
in	<i>SampleLocation</i>	An absolute sample location (see GetCurrentNativeSampleLocation())

Implemented in [AAX_VTransport](#).

14.132.3.9 GetTicksPerQuarter()

```
virtual AAX_Result AAX_ITransport::GetTicksPerQuarter (
    uint32_t * ticks ) const [pure virtual]
```

CALL: Retrieves the number of ticks per quarter note.

Parameters

out	<i>ticks</i>	
-----	--------------	--

Implemented in [AAX_VTransport](#).

14.132.3.10 GetCurrentTicksPerBeat()

```
virtual AAX_Result AAX_ITransport::GetCurrentTicksPerBeat (
    uint32_t * ticks ) const [pure virtual]
```

CALL: Retrieves the number of ticks per beat.

Parameters

out	<i>ticks</i>	
-----	--------------	--

Implemented in [AAX_VTransport](#).

14.132.3.11 GetTimelineSelectionStartPosition()

```
virtual AAX_Result AAX_ITransport::GetTimelineSelectionStartPosition (
    int64_t * oSampleLocation ) const [pure virtual]
```

CALL: Retrieves the absolute sample position of the beginning of the current transport selection.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oSampleLocation</i>	
-----	------------------------	--

Implemented in [AAX_VTransport](#).

14.132.3.12 GetTimeCodeInfo()

```
virtual AAX_Result AAX_ITransport::GetTimeCodeInfo (
    AAX_EFrameRate * oFrameRate,
    int32_t * oOffset ) const [pure virtual]
```

CALL: Retrieves the current time code frame rate and offset.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oFrameRate</i>	
out	<i>oOffset</i>	

Implemented in [AAX_VTransport](#).

14.132.3.13 GetFeetFramesInfo()

```
virtual AAX_Result AAX_ITransport::GetFeetFramesInfo (
    AAX_EFeetFramesRate * oFeetFramesRate,
    int64_t * oOffset ) const [pure virtual]
```

CALL: Retrieves the current timecode feet/frames rate and offset.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oFeetFramesRate</i>	
out	<i>oOffset</i>	

Implemented in [AAX_VTransport](#).

14.132.3.14 IsMetronomeEnabled()

```
virtual AAX_Result AAX_ITransport::IsMetronomeEnabled (
    int32_t * isEnabled ) const [pure virtual]
```

Sets isEnabled to true if the metronome is enabled.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>isEnabled</i>	
-----	------------------	--

Implemented in [AAX_VTransport](#).

14.132.3.15 GetHDTimeCodeInfo()

```
virtual AAX_Result AAX_ITransport::GetHDTimeCodeInfo (
    AAX_EFrameRate * oHDFrameRate,
    int64_t * oHDOffset ) const [pure virtual]
```

CALL: Retrieves the current HD time code frame rate and offset.

Note

This method is part of the [version 3 transport interface](#)

Parameters

out	<i>oHDFrameRate</i>	
out	<i>oHDOffset</i>	

Implemented in [AAX_VTransport](#).

14.132.3.16 RequestTransportStart()

```
virtual AAX\_Result AAX_ITransport::RequestTransportStart ( ) [pure virtual]
```

CALL: Request that the host transport start playback.

Note

This method is part of the [AAX_IACFTransportControl](#) interface

Implemented in [AAX_VTransport](#).

14.132.3.17 RequestTransportStop()

```
virtual AAX\_Result AAX_ITransport::RequestTransportStop ( ) [pure virtual]
```

CALL: Request that the host transport stop playback.

Note

This method is part of the [AAX_IACFTransportControl](#) interface

Implemented in [AAX_VTransport](#).

14.132.3.18 GetTimelineSelectionEndPosition()

```
virtual AAX\_Result AAX_ITransport::GetTimelineSelectionEndPosition (
    int64_t * oSampleLocation ) const [pure virtual]
```

CALL: Retrieves the absolute sample position of the end of the current transport selection.

Note

This method is part of the [version 4 transport interface](#)

Parameters

out	<i>oSampleLocation</i>	
-----	------------------------	--

Implemented in [AAX_VTransport](#).

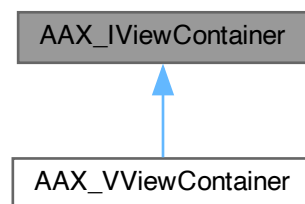
The documentation for this class was generated from the following file:

- [AAX_ITransport.h](#)

14.133 AAX_IViewContainer Class Reference

```
#include <AAX_IViewContainer.h>
```

Inheritance diagram for AAX_IViewContainer:



14.133.1 Description

Interface for the AAX host's view of a single instance of an effect. Used both by clients of the AAX host and by effect components.

:Implemented by the AAX Host

Public Member Functions

- virtual [~AAX_IViewContainer](#) (void)

View and GUI state queries

- virtual `int32_t` [GetType](#) ()=0
Returns the raw view type as one of [AAX_EViewContainer_Type](#).
- virtual `void *` [GetPtr](#) ()=0
Returns a pointer to the raw view.
- virtual [AAX_Result](#) [GetModifiers](#) (uint32_t *outModifiers)=0

Queries the host for the current [modifier keys](#).

View change requests

- virtual [AAX_Result SetViewSize](#) ([AAX_Point](#) &inSize)=0
Request a change to the main view size.

Host event handlers

These methods are used to pass plug-in GUI events to the host for handling. Events should always be passed on in this way when there is a possibility of the host overriding the event with its own behavior.

For example, in *Pro Tools* a command-control-option-click on any automatable plug-in parameter editor should bring up that parameter's automation pop-up menu, and a control-right click should display the parameter's automation lane in the *Pro Tools* Edit window. In order for *Pro Tools* to handle these events, the plug-in must pass them on using [HandleParameterMouseDown\(\)](#)

For each of these methods:

- [AAX_SUCCESS](#) is returned if the event was successfully handled by the host. In most cases, no further action will be required from the plug-in after the host successfully handles an event.
- [AAX_ERROR_UNIMPLEMENTED](#) is returned if the event was not handled by the host. In this case, the plug-in should perform its own event handling.
- virtual [AAX_Result HandleParameterMouseDown](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse down event.
- virtual [AAX_Result HandleParameterMouseDrag](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse drag event.
- virtual [AAX_Result HandleParameterMouseUp](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse up event.
- virtual [AAX_Result HandleParameterMouseEnter](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse enter event to the parameter's control.
- virtual [AAX_Result HandleParameterMouseExit](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers)=0
Alert the host to a mouse exit event from the parameter's control.
- virtual [AAX_Result HandleMultipleParametersMouseDown](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse down event.
- virtual [AAX_Result HandleMultipleParametersMouseDrag](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse drag event.
- virtual [AAX_Result HandleMultipleParametersMouseUp](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers)=0
Alert the host to a mouse up event.

14.133.2 Constructor & Destructor Documentation

14.133.2.1 ~AAX_IViewContainer()

```
virtual AAX_IViewContainer::~~AAX_IViewContainer (
    void ) [inline], [virtual]
```

14.133.3 Member Function Documentation

14.133.3.1 GetType()

```
virtual int32_t AAX_IViewContainer::GetType ( ) [pure virtual]
```

Returns the raw view type as one of [AAX_EViewContainer_Type](#).

Implemented in [AAX_VViewContainer](#).

14.133.3.2 GetPtr()

```
virtual void * AAX_IViewContainer::GetPtr ( ) [pure virtual]
```

Returns a pointer to the raw view.

Implemented in [AAX_VViewContainer](#).

14.133.3.3 GetModifiers()

```
virtual AAX\_Result AAX_IViewContainer::GetModifiers (
    uint32_t * outModifiers ) [pure virtual]
```

Queries the host for the current [modifier keys](#).

This method returns a bit mask with bits set for each of the currently active modifier keys. This method does not return the state of the [AAX_eModifiers_SecondaryButton](#).

Host Compatibility Notes Although this method allows plug-ins to acquire the current state of the Windows key (normally blocked by Pro Tools), plug-ins should not use key combinations that require this key.

Parameters

out	<i>outModifiers</i>	Current modifiers as a bitmask of AAX_EModifiers
-----	---------------------	--

Implemented in [AAX_VViewContainer](#).

14.133.3.4 SetViewSize()

```
virtual AAX_Result AAX_IViewContainer::SetViewSize (
    AAX_Point & inSize ) [pure virtual]
```

Request a change to the main view size.

Note

- For compatibility with the smallest supported displays, plug-in GUI dimensions should not exceed 749x617 pixels, or 749x565 pixels for plug-ins with sidechain support.

Parameters

in	<i>inSize</i>	The new size to which the plug-in view should be set
----	---------------	--

Implemented in [AAX_VViewContainer](#).

14.133.3.5 HandleParameterMouseDown()

```
virtual AAX_Result AAX_IViewContainer::HandleParameterMouseDown (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse down event.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implemented in [AAX_VViewContainer](#).

14.133.3.6 HandleParameterMouseDrag()

```
virtual AAX_Result AAX_IViewContainer::HandleParameterMouseDrag (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse drag event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implemented in [AAX_VViewContainer](#).

14.133.3.7 HandleParameterMouseUp()

```
virtual AAX\_Result AAX_IViewContainer::HandleParameterMouseUp (
    AAX\_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse up event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implemented in [AAX_VViewContainer](#).

14.133.3.8 HandleParameterMouseEnter()

```
virtual AAX\_Result AAX_IViewContainer::HandleParameterMouseEnter (
    AAX\_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse enter event to the parameter's control.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being entered
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Returns [AAX_SUCCESS](#) if event was processed successfully, otherwise an [AAX_ERROR](#) code

Implemented in [AAX_VViewContainer](#).

14.133.3.9 HandleParameterMouseExit()

```
virtual AAX_Result AAX_IViewContainer::HandleParameterMouseExit (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse exit event from the parameter's control.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being exited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Returns [AAX_SUCCESS](#) if event was processed successfully, otherwise an [AAX_ERROR](#) code

Implemented in [AAX_VViewContainer](#).

14.133.3.10 HandleMultipleParametersMouseDown()

```
virtual AAX_Result AAX_IViewContainer::HandleMultipleParametersMouseDown (
    const AAX_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse down event.

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implemented in [AAX_VViewContainer](#).

14.133.3.11 HandleMultipleParametersMouseDrag()

```
virtual AAX_Result AAX_IViewContainer::HandleMultipleParametersMouseDrag (
    const AAX_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse drag event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implemented in [AAX_VViewContainer](#).

14.133.3.12 HandleMultipleParametersMouseUp()

```
virtual AAX\_Result AAX_IViewContainer::HandleMultipleParametersMouseUp (
    const AAX\_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [pure virtual]
```

Alert the host to a mouse up event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implemented in [AAX_VViewContainer](#).

The documentation for this class was generated from the following file:

- [AAX_IViewContainer.h](#)

14.134 AAX_Map Class Reference

```
#include <AAX_Map.h>
```


Public Member Functions

- [AAX_Map](#) ()
- [~AAX_Map](#) ()
- void [SetCoefficients](#) (int aSize, double *aInpX, double *aInpY)
- void [GetCoefficient](#) (int aIndex, double *aOutX, double *aOutY)
- int [GetUpperBoundIndex](#) (double inp)
- double [GetX](#) (int aIndex)
- double [GetY](#) (int aIndex)
- double [GetFirstX](#) ()
- double [GetFirstY](#) ()
- double [GetLastX](#) ()
- double [GetLastY](#) ()
- int [GetSize](#) ()

14.134.1 Constructor & Destructor Documentation

14.134.1.1 AAX_Map()

```
AAX_Map::AAX_Map ( ) [inline]
```

14.134.1.2 ~AAX_Map()

```
AAX_Map::~~AAX_Map ( ) [inline]
```

14.134.2 Member Function Documentation

14.134.2.1 SetCoefficients()

```
void AAX_Map::SetCoefficients (
    int aSize,
    double * aInpX,
    double * aInpY )
```

14.134.2.2 GetCoefficient()

```
void AAX_Map::GetCoefficient (
    int aIndex,
    double * aOutX,
    double * aOutY )
```

14.134.2.3 GetUpperBoundIndex()

```
int AAX_Map::GetUpperBoundIndex (
    double inp )
```

14.134.2.4 GetX()

```
double AAX_Map::GetX (
    int aIndex ) [inline]
```

14.134.2.5 GetY()

```
double AAX_Map::GetY (
    int aIndex ) [inline]
```

14.134.2.6 GetFirstX()

```
double AAX_Map::GetFirstX ( ) [inline]
```

14.134.2.7 GetFirstY()

```
double AAX_Map::GetFirstY ( ) [inline]
```

14.134.2.8 GetLastX()

```
double AAX_Map::GetLastX ( ) [inline]
```

14.134.2.9 GetLastY()

```
double AAX_Map::GetLastY ( ) [inline]
```

14.134.2.10 GetSize()

```
int AAX_Map::GetSize ( ) [inline]
```

The documentation for this class was generated from the following file:

- [AAX_Map.h](#)

14.135 AAX_Point Struct Reference

```
#include <AAX_GUITypes.h>
```

14.135.1 Description

Data structure representing a two-dimensional coordinate point.

Comparison operators give preference to `vert`

Public Member Functions

- [AAX_Point](#) (float v, float h)
- [AAX_Point](#) (void)

Public Attributes

- float [vert](#)
- float [horz](#)

14.135.2 Constructor & Destructor Documentation

14.135.2.1 AAX_Point() [1/2]

```
AAX_Point::AAX_Point (
    float v,
    float h ) [inline]
```

14.135.2.2 AAX_Point() [2/2]

```
AAX_Point::AAX_Point (
    void ) [inline]
```

14.135.3 Member Data Documentation

14.135.3.1 vert

`float AAX_Point::vert`

Referenced by [operator<\(\)](#), [operator<=\(\)](#), and [operator==\(\)](#).

14.135.3.2 horz

`float AAX_Point::horz`

Referenced by [operator<\(\)](#), [operator<=\(\)](#), and [operator==\(\)](#).

The documentation for this struct was generated from the following file:

- [AAX_GUITypes.h](#)

14.136 AAX_Rect Struct Reference

```
#include <AAX_GUITypes.h>
```

14.136.1 Description

Data structure representing a rectangle in a two-dimensional coordinate plane.

Public Member Functions

- [AAX_Rect](#) (float t, float l, float w, float h)
- [AAX_Rect](#) (void)

Public Attributes

- float [top](#)
- float [left](#)
- float [width](#)
- float [height](#)

14.136.2 Constructor & Destructor Documentation

14.136.2.1 AAX_Rect() [1/2]

```
AAX_Rect::AAX_Rect (
    float t,
    float l,
    float w,
    float h ) [inline]
```

14.136.2.2 AAX_Rect() [2/2]

```
AAX_Rect::AAX_Rect (
    void ) [inline]
```

14.136.3 Member Data Documentation

14.136.3.1 top

```
float AAX_Rect::top
```

Referenced by [operator==\(.\)](#).

14.136.3.2 left

```
float AAX_Rect::left
```

Referenced by [operator==\(.\)](#).

14.136.3.3 width

```
float AAX_Rect::width
```

Referenced by [operator==\(.\)](#).

14.136.3.4 height

```
float AAX_Rect::height
```

Referenced by [operator==\(\)](#).

The documentation for this struct was generated from the following file:

- [AAX_GUITypes.h](#)

14.137 AAX_SHybridRenderInfo Struct Reference

```
#include <AAX_IACFEEffectParameters.h>
```

14.137.1 Description

Hybrid render processing context.

See also

[AAX_IACFEEffectParameters_V2::RenderAudio_Hybrid\(\)](#)

Public Attributes

- float ** [mAudioInputs](#)
- int32_t * [mNumAudioInputs](#)
- float ** [mAudioOutputs](#)
- int32_t * [mNumAudioOutputs](#)
- int32_t * [mNumSamples](#)
- [AAX_CTimestamp](#) * [mClock](#)

14.137.2 Member Data Documentation

14.137.2.1 mAudioInputs

```
float** AAX_SHybridRenderInfo::mAudioInputs
```

14.137.2.2 mNumAudioInputs

```
int32_t* AAX_SHybridRenderInfo::mNumAudioInputs
```


Public Attributes

- float ** [mAudioInputs](#)
Audio input buffers.
- float ** [mAudioOutputs](#)
Audio output buffers, including any aux output stems.
- int32_t * [mNumSamples](#)
Number of samples in each buffer. Bounded as per [AAE_EAudioBufferLengthNative](#). The exact value can vary from buffer to buffer.
- [AAX_CTimestamp](#) * [mClock](#)
Pointer to the global running time clock.
- [AAX_IMIDINode](#) * [mInputNode](#)
Buffered local MIDI input node. Used for incoming MIDI messages directed to the instrument.
- [AAX_IMIDINode](#) * [mGlobalNode](#)
Buffered global MIDI input node. Used for global events like beat updates in metronomes.
- [AAX_IMIDINode](#) * [mTransportNode](#)
Transport MIDI node. Used for querying the state of the MIDI transport.
- [AAX_IMIDINode](#) * [mAdditionalInputMIDINodes](#) [[kMaxAdditionalMIDINodes](#)]
List of additional input MIDI nodes, if your plugin needs them.
- [AAX_SInstrumentPrivateData](#) * [mPrivateData](#)
Struct containing private data relating to the instance. You should not need to use this data.
- float ** [mMeters](#)
Array of meter taps. One meter value should be entered per tap for each render call.
- int64_t * [mCurrentStateNum](#)
State counter.

14.139.2 Member Data Documentation

14.139.2.1 mAudioInputs

```
float** AAX_SInstrumentRenderInfo::mAudioInputs
```

Audio input buffers.

14.139.2.2 mAudioOutputs

```
float** AAX_SInstrumentRenderInfo::mAudioOutputs
```

Audio output buffers, including any aux output stems.

14.139.2.3 mNumSamples

```
int32_t* AAX_SInstrumentRenderInfo::mNumSamples
```

Number of samples in each buffer. Bounded as per [AAE_EAudioBufferLengthNative](#). The exact value can vary from buffer to buffer.

14.139.2.4 mClock

```
AAX_CTimestamp* AAX_SInstrumentRenderInfo::mClock
```

Pointer to the global running time clock.

14.139.2.5 mInputNode

```
AAX_IMIDIINode* AAX_SInstrumentRenderInfo::mInputNode
```

Buffered local MIDI input node. Used for incoming MIDI messages directed to the instrument.

14.139.2.6 mGlobalNode

```
AAX_IMIDIINode* AAX_SInstrumentRenderInfo::mGlobalNode
```

Buffered global MIDI input node. Used for global events like beat updates in metronomes.

14.139.2.7 mTransportNode

```
AAX_IMIDIINode* AAX_SInstrumentRenderInfo::mTransportNode
```

Transport MIDI node. Used for querying the state of the MIDI transport.

14.139.2.8 mAdditionalInputMIDINodes

```
AAX_IMIDIINode* AAX_SInstrumentRenderInfo::mAdditionalInputMIDINodes[kMaxAdditionalMIDINodes]
```

List of additional input MIDI nodes, if your plugin needs them.

14.139.2.9 mPrivateData

[AAX_SInstrumentPrivateData*](#) AAX_SInstrumentRenderInfo::mPrivateData

Struct containing private data relating to the instance. You should not need to use this data.

14.139.2.10 mMeters

float** AAX_SInstrumentRenderInfo::mMeters

Array of meter taps. One meter value should be entered per tap for each render call.

14.139.2.11 mCurrentStateNum

int64_t* AAX_SInstrumentRenderInfo::mCurrentStateNum

State counter.

The documentation for this struct was generated from the following file:

- [AAX_CMonolithicParameters.h](#)

14.140 AAX_SInstrumentSetupInfo Struct Reference

```
#include <AAX_CMonolithicParameters.h>
```

14.140.1 Description

Information used to describe the instrument.

See also

[AAX_CMonolithicParameters::StaticDescribe\(\)](#)

Public Member Functions

- [AAX_SInstrumentSetupInfo \(\)](#)

Default constructor.

Public Attributes

- bool [mNeedsGlobalMIDI](#)
Does the instrument use a global MIDI input node?
- const char * [mGlobalMIDINodeName](#)
Name of the global MIDI node, if used.
- uint32_t [mGlobalMIDIEventMask](#)
Global MIDI node event mask of [AAX_EMidGlobalNodeSelectors](#), if used.
- bool [mNeedsInputMIDI](#)
Does the instrument use a local MIDI input node?
- const char * [mInputMIDINodeName](#)
Name of the MIDI input node, if used.
- uint32_t [mInputMIDIChannelMask](#)
MIDI input node channel mask, if used.
- int32_t [mNumAdditionalInputMIDINodes](#)
Number of additional input MIDI Nodes. These will all share the same channelMask and base MIDINodeName, but the names will be appended with numbers 2,3,4,...
- bool [mNeedsTransport](#)
Does the instrument use the transport interface?
- const char * [mTransportMIDINodeName](#)
Name of the MIDI transport node, if used.
- int32_t [mNumMeters](#)
Number of meter taps used by the instrument. Must match the size of [mMeterIDs](#).
- const [AAX_CTypeID](#) * [mMeterIDs](#)
Array of meter IDs.
- int32_t [mNumAuxOutputStems](#)
Number of aux output stems for the plug-in.
- const char * [mAuxOutputStemNames](#) [[kMaxAuxOutputStems](#)]
Names of the aux output stems.
- [AAX_EStemFormat](#) [mAuxOutputStemFormats](#) [[kMaxAuxOutputStems](#)]
Stem formats for the output stems.
- [AAX_EStemFormat](#) [mHybridInputStemFormat](#)
Hybrid input stem format
- [AAX_EStemFormat](#) [mHybridOutputStemFormat](#)
Hybrid output stem format
- [AAX_EStemFormat](#) [mInputStemFormat](#)
Input stem format
- [AAX_EStemFormat](#) [mOutputStemFormat](#)
Output stem format
- bool [mUseHostGeneratedGUI](#)
Allow Pro Tools or other host to generate a generic GUI. This can be useful for early development.
- bool [mCanBypass](#)
Can this instrument be bypassed?
- [AAX_CTypeID](#) [mManufacturerID](#)
Manufacturer ID
- [AAX_CTypeID](#) [mProductID](#)
Product ID
- [AAX_CTypeID](#) [mPluginID](#)
Plug-In (Type) ID
- [AAX_CTypeID](#) [mAudiosuiteID](#)
AudioSuite ID
- [AAX_CBoolean](#) [mMultiMonoSupport](#)

14.140.2 Constructor & Destructor Documentation

14.140.2.1 AAX_SInstrumentSetupInfo()

```
AAX_SInstrumentSetupInfo::AAX_SInstrumentSetupInfo ( ) [inline]
```

Default constructor.

Use this constructor if you want to enable a sub-set of features and don't need to fill out the whole struct.

References [AAX_eStemFormat_Mono](#), [AAX_eStemFormat_None](#), [kMaxAuxOutputStems](#), [mAudiosuiteID](#), [mAuxOutputStemFormats](#), [mAuxOutputStemNames](#), [mCanBypass](#), [mGlobalMIDIEventMask](#), [mGlobalMIDINodeName](#), [mHybridInputStemFormat](#), [mHybridOutputStemFormat](#), [mInputMIDIChannelMask](#), [mInputMIDINodeName](#), [mInputStemFormat](#), [mManufacturerID](#), [mMeterIDs](#), [mMultiMonoSupport](#), [mNeedsGlobalMIDI](#), [mNeedsInputMIDI](#), [mNeedsTransport](#), [mNumAdditionalInputMIDINodes](#), [mNumAuxOutputStems](#), [mNumMeters](#), [mOutputStemFormat](#), [mPluginID](#), [mProductID](#), [mTransportMIDINodeName](#), and [mUseHostGeneratedGUI](#).

14.140.3 Member Data Documentation

14.140.3.1 mNeedsGlobalMIDI

```
bool AAX_SInstrumentSetupInfo::mNeedsGlobalMIDI
```

Does the instrument use a global MIDI input node?

See also

[MIDI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.2 mGlobalMIDINodeName

```
const char* AAX_SInstrumentSetupInfo::mGlobalMIDINodeName
```

Name of the global MIDI node, if used.

See also

[MIDI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.3 mGlobalMIDIEventMask

```
uint32_t AAX_SInstrumentSetupInfo::mGlobalMIDIEventMask
```

Global MIDI node event mask of [AAX_EMidiGlobalNodeSelectors](#), if used.

See also

[MIDI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.4 mNeedsInputMIDI

```
bool AAX_SInstrumentSetupInfo::mNeedsInputMIDI
```

Does the instrument use a local MIDI input node?

See also

[MIDI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.5 mInputMIDINodeName

```
const char* AAX_SInstrumentSetupInfo::mInputMIDINodeName
```

Name of the MIDI input node, if used.

See also

[MIDI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.6 mInputMIDIChannelMask

```
uint32_t AAX_SInstrumentSetupInfo::mInputMIDIChannelMask
```

MIDI input node channel mask, if used.

See also

[MIDI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.7 mNumAdditionalInputMIDINodes

```
int32_t AAX_SInstrumentSetupInfo::mNumAdditionalInputMIDINodes
```

Number of additional input MIDI Nodes. These will all share the same channelMask and base MIDINodeName, but the names will be appended with numbers 2,3,4,...

See also

[MIDI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.8 mNeedsTransport

```
bool AAX_SInstrumentSetupInfo::mNeedsTransport
```

Does the instrument use the transport interface?

See also

[AAX_ITransport](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.9 mTransportMIDINodeName

```
const char* AAX_SInstrumentSetupInfo::mTransportMIDINodeName
```

Name of the MIDI transport node, if used.

See also

[AAX_ITransport](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#).

14.140.3.10 mNumMeters

```
int32_t AAX_SInstrumentSetupInfo::mNumMeters
```

Number of meter taps used by the instrument. Must match the size of [mMeterIDs](#).

See also

[Plug-in meters](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.11 mMeterIDs

```
const AAX_CTypeID* AAX_SInstrumentSetupInfo::mMeterIDs
```

Array of meter IDs.

See also

[Plug-in meters](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.12 mNumAuxOutputStems

```
int32_t AAX_SInstrumentSetupInfo::mNumAuxOutputStems
```

Number of aux output stems for the plug-in.

See also

[Auxiliary Output Stems](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.13 mAuxOutputStemNames

```
const char* AAX_SInstrumentSetupInfo::mAuxOutputStemNames[kMaxAuxOutputStems]
```

Names of the aux output stems.

See also

[Auxiliary Output Stems](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.14 mAuxOutputStemFormats

```
AAX_EStemFormat AAX_SInstrumentSetupInfo::mAuxOutputStemFormats[kMaxAuxOutputStems]
```

Stem formats for the output stems.

See also

[Auxiliary Output Stems](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.15 mHybridInputStemFormat

[AAX_EStemFormat](#) AAX_SInstrumentSetupInfo::mHybridInputStemFormat

Hybrid input stem format

A plug-in that defines this value must also define [mHybridOutputStemFormat](#) and implement [AAX_IEffectParameters::RenderAudio_Hybrid\(\)](#)

See also

[Hybrid Processing architecture](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.16 mHybridOutputStemFormat

[AAX_EStemFormat](#) AAX_SInstrumentSetupInfo::mHybridOutputStemFormat

Hybrid output stem format

A plug-in that defines this value must also define [mHybridInputStemFormat](#) and implement [AAX_IEffectParameters::RenderAudio_Hybrid\(\)](#)

See also

[Hybrid Processing architecture](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.17 mInputStemFormat

[AAX_EStemFormat](#) AAX_SInstrumentSetupInfo::mInputStemFormat

Input stem format

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.18 mOutputStemFormat

[AAX_EStemFormat](#) AAX_SInstrumentSetupInfo::mOutputStemFormat

Output stem format

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.19 mUseHostGeneratedGUI

```
bool AAX_SInstrumentSetupInfo::mUseHostGeneratedGUI
```

Allow Pro Tools or other host to generate a generic GUI. This can be useful for early development.

See also

[AAX_eProperty_UsesClientGUI](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.20 mCanBypass

```
bool AAX_SInstrumentSetupInfo::mCanBypass
```

Can this instrument be bypassed?

See also

[AAX_eProperty_CanBypass](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.21 mManufacturerID

```
AAX_CTypeID AAX_SInstrumentSetupInfo::mManufacturerID
```

[Manufacturer ID](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.22 mProductID

```
AAX_CTypeID AAX_SInstrumentSetupInfo::mProductID
```

[Product ID](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.23 mPluginID

[AAX_CTypeID](#) `AAX_SInstrumentSetupInfo::mPluginID`

Plug-In (Type) ID

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.24 mAudiosuiteID

[AAX_CTypeID](#) `AAX_SInstrumentSetupInfo::mAudiosuiteID`

AudioSuite ID

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

14.140.3.25 mMultiMonoSupport

[AAX_CBoolean](#) `AAX_SInstrumentSetupInfo::mMultiMonoSupport`

Multi-mono support

Note

It is recommended to un-set the `mMultiMonoSupport` flag for VIs and other plug-ins which rely on non-global MIDI input. For more information see [AAX_eProperty_Constraint_MultiMonoSupport](#)

Referenced by [AAX_SInstrumentSetupInfo\(\)](#), and [AAX_CMonolithicParameters::StaticDescribe\(\)](#).

The documentation for this struct was generated from the following file:

- [AAX_CMonolithicParameters.h](#)

14.141 AAX_SPlugInChunk Struct Reference

```
#include <AAX.h>
```

14.141.1 Description

Plug-in chunk header + data.

See also

[AAX_SPlugInChunkHeader](#)

Public Attributes

- `int32_t fSize`
The size of the chunk's [fData](#) member.
- `int32_t fVersion`
The chunk's version.
- `AAX_CTypeID fManufacturerID`
The Plug-In's manufacturer ID.
- `AAX_CTypeID fProductID`
The Plug-In file's product ID.
- `AAX_CTypeID fPlugInID`
The ID of a particular Plug-In within the file.
- `AAX_CTypeID fChunkID`
The ID of a particular Plug-In chunk.
- `unsigned char fName [32]`
A user defined name for this chunk.
- `char fData [1]`
The chunk's data.

14.141.2 Member Data Documentation

14.141.2.1 fSize

```
int32_t AAX_SPlugInChunk::fSize
```

The size of the chunk's [fData](#) member.

14.141.2.2 fVersion

```
int32_t AAX_SPlugInChunk::fVersion
```

The chunk's version.

14.141.2.3 fManufacturerID

```
AAX\_CTypeID AAX_SPlugInChunk::fManufacturerID
```

The Plug-In's manufacturer ID.

14.141.2.4 fProductID

[AAX_CTypeID](#) AAX_SPlugInChunk::fProductID

The Plug-In file's product ID.

14.141.2.5 fPlugInID

[AAX_CTypeID](#) AAX_SPlugInChunk::fPlugInID

The ID of a particular Plug-In within the file.

14.141.2.6 fChunkID

[AAX_CTypeID](#) AAX_SPlugInChunk::fChunkID

The ID of a particular Plug-In chunk.

14.141.2.7 fName

unsigned char AAX_SPlugInChunk::fName[32]

A user defined name for this chunk.

14.141.2.8 fData

char AAX_SPlugInChunk::fData[1]

The chunk's data.

Note

The fixed-size array definition here is historical, but misleading. Plug-ins actually write off the end of this block and are allowed to as long as they don't exceed their reported size.

The documentation for this struct was generated from the following file:

- [AAX.h](#)

14.142 AAX_SPlugInChunkHeader Struct Reference

```
#include <AAX.h>
```

14.142.1 Description

Plug-in chunk header.

Legacy Porting Notes To ensure compatibility with TDM/RTAS plug-ins whose implementation requires `fSize` to be equal to the size of the chunk's header plus its data, AAE performs some behind-the-scenes record keeping.

The following actions are only taken for AAX plug-ins, so, e.g., if a chunk is stored by an RTAS or TDM plug-in that reports data+header size in `fSize` and this chunk is then loaded by the AAX version of the plug-in, the header size will be cached as-is from the legacy plug-in and will be subtracted out before the chunk data is passed to the AAX plug-in. If a chunk is stored by an AAX plug-in and is then loaded by a legacy plug-in, the legacy plug-in will receive the cached plug-in header with `fSize` equal to the data+header size.

These are the special actions that AAE takes to ensure backwards-compatibility when handling AAX chunk data:

- When AAE retrieves the size of a chunk from an AAX plug-in using `GetChunkSize()`, it adds the chunk header size to the amount of memory that it allocates for the chunk
- When AAE retrieves a chunk from an AAX plug-in using `GetChunk()`, it adds the chunk header size to `fChunkSize` before caching the chunk
- Before calling `SetChunk()` or `CompareActiveChunk()`, AAE subtracts the chunk header size from the cached chunk's header's `fChunkSize` member

Public Attributes

- `int32_t fSize`
The size of the chunk's `fData` member.
- `int32_t fVersion`
The chunk's version.
- `AAX_CTypeID fManufacturerID`
The Plug-In's manufacturer ID.
- `AAX_CTypeID fProductID`
The Plug-In file's product ID.
- `AAX_CTypeID fPlugInID`
The ID of a particular Plug-In within the file.
- `AAX_CTypeID fChunkID`
The ID of a particular Plug-In chunk.
- `unsigned char fName [32]`
A user defined name for this chunk.

14.142.2 Member Data Documentation

14.142.2.1 fSize

```
int32_t AAX_SPlugInChunkHeader::fSize
```

The size of the chunk's [fData](#) member.

14.142.2.2 fVersion

```
int32_t AAX_SPlugInChunkHeader::fVersion
```

The chunk's version.

14.142.2.3 fManufacturerID

```
AAX\_CTypeID AAX_SPlugInChunkHeader::fManufacturerID
```

The Plug-In's manufacturer ID.

14.142.2.4 fProductID

```
AAX\_CTypeID AAX_SPlugInChunkHeader::fProductID
```

The Plug-In file's product ID.

14.142.2.5 fPlugInID

```
AAX\_CTypeID AAX_SPlugInChunkHeader::fPlugInID
```

The ID of a particular Plug-In within the file.

14.142.2.6 fChunkID

[AAX_CTypeID](#) AAX_SPlugInChunkHeader::fChunkID

The ID of a particular Plug-In chunk.

14.142.2.7 fName

unsigned char AAX_SPlugInChunkHeader::fName[32]

A user defined name for this chunk.

The documentation for this struct was generated from the following file:

- [AAX.h](#)

14.143 AAX_SPlugInIdentifierTriad Struct Reference

```
#include <AAX.h>
```

14.143.1 Description

Plug-in Identifier Triad.

This set of identifiers are what uniquely identify a particular plug-in type.

Public Attributes

- [AAX_CTypeID](#) mManufacturerID
The Plug-In's manufacturer ID.
- [AAX_CTypeID](#) mProductID
The Plug-In's product (Effect) ID.
- [AAX_CTypeID](#) mPlugInID
The ID of a specific type in the product (Effect)

14.143.2 Member Data Documentation

14.143.2.1 mManufacturerID

[AAX_CTypeID](#) AAX_SPlugInIdentifierTriad::mManufacturerID

The Plug-In's manufacturer ID.

Referenced by [AAX::AsStringIDTriad\(\)](#).

14.143.2.2 mProductID

[AAX_CTypeID](#) AAX_SPlugInIdentifierTriad::mProductID

The Plug-In's product (Effect) ID.

Referenced by [AAX::AsStringIDTriad\(\)](#).

14.143.2.3 mPlugInID

[AAX_CTypeID](#) AAX_SPlugInIdentifierTriad::mPlugInID

The ID of a specific type in the product (Effect)

Referenced by [AAX::AsStringIDTriad\(\)](#).

The documentation for this struct was generated from the following file:

- [AAX.h](#)

14.144 AAX_StLock_Guard Class Reference

```
#include <AAX_CMutex.h>
```

14.144.1 Description

Helper class for working with mutex.

Public Member Functions

- [AAX_StLock_Guard](#) ([AAX_CMutex](#) &iMutex)
- [~AAX_StLock_Guard](#) ()

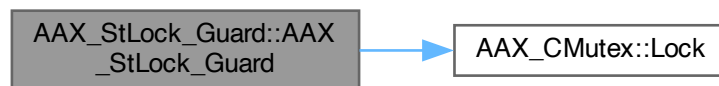
14.144.2 Constructor & Destructor Documentation

14.144.2.1 AAX_StLock_Guard()

```
AAX_StLock_Guard::AAX_StLock_Guard (
    AAX_CMutex & iMutex ) [inline], [explicit]
```

References [AAX_CMutex::Lock\(\)](#).

Here is the call graph for this function:

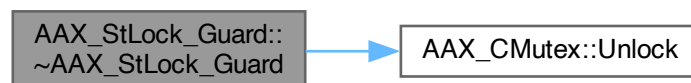


14.144.2.2 ~AAX_StLock_Guard()

```
AAX_StLock_Guard::~~AAX_StLock_Guard ( ) [inline]
```

References [AAX_CMutex::Unlock\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [AAX_CMutex.h](#)

14.145 AAX_TransportStateInfo_V1 Struct Reference

```
#include <AAX_TransportTypes.h>
```

14.145.1 Description

Helper structure for payload data described transport state information.

Public Member Functions

- [AAX_TransportStateInfo_V1](#) ()
- `std::string ToString () const`

Public Attributes

- [AAX_ETransportState](#) mTransportState
- [AAX_ERecordMode](#) mRecordMode
- [AAX_CBoolean](#) mIsRecordEnabled
- [AAX_CBoolean](#) mIsRecording
- [AAX_CBoolean](#) mIsLoopEnabled

14.145.2 Constructor & Destructor Documentation

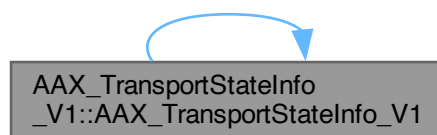
14.145.2.1 AAX_TransportStateInfo_V1()

```
AAX_TransportStateInfo_V1::AAX_TransportStateInfo_V1 ( ) [inline]
```

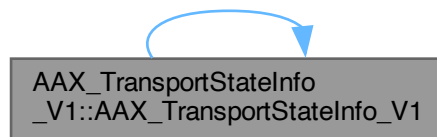
References [AAX_TransportStateInfo_V1\(\)](#).

Referenced by [AAX_TransportStateInfo_V1\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.145.3 Member Function Documentation

14.145.3.1 ToString()

```
std::string AAX_TransportStateInfo_V1::ToString ( ) const [inline]
```

References [mIsLoopEnabled](#), [mIsRecordEnabled](#), [mIsRecording](#), [mRecordMode](#), and [mTransportState](#).

14.145.4 Member Data Documentation

14.145.4.1 mTransportState

```
AAX_ETransportState AAX_TransportStateInfo_V1::mTransportState
```

Referenced by [operator==\(\)](#), and [ToString\(\)](#).

14.145.4.2 mRecordMode

```
AAX_ERecordMode AAX_TransportStateInfo_V1::mRecordMode
```

Referenced by [operator==\(\)](#), and [ToString\(\)](#).

14.145.4.3 mIsRecordEnabled

`AAX_CBoolean AAX_TransportStateInfo_V1::mIsRecordEnabled`

Referenced by `operator==()`, and `ToString()`.

14.145.4.4 mIsRecording

`AAX_CBoolean AAX_TransportStateInfo_V1::mIsRecording`

Referenced by `operator==()`, and `ToString()`.

14.145.4.5 mIsLoopEnabled

`AAX_CBoolean AAX_TransportStateInfo_V1::mIsLoopEnabled`

Referenced by `operator==()`, and `ToString()`.

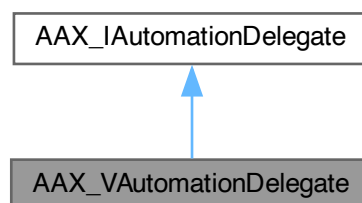
The documentation for this struct was generated from the following file:

- [AAX_TransportTypes.h](#)

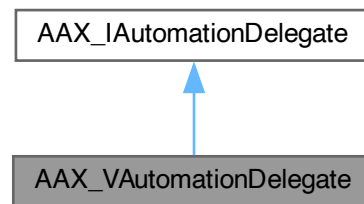
14.146 AAX_VAutomationDelegate Class Reference

```
#include <AAX_VAutomationDelegate.h>
```

Inheritance diagram for AAX_VAutomationDelegate:



Collaboration diagram for AAX_VAutomationDelegate:



14.146.1 Description

Version-managed concrete [automation delegate](#) class.

Public Member Functions

- [AAX_VAutomationDelegate](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VAutomationDelegate](#) () [AAX_OVERRIDE](#)
- [IACFUnknown](#) * [GetUnknown](#) () const
- [AAX_Result](#) [RegisterParameter](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)
- [AAX_Result](#) [UnregisterParameter](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)
- [AAX_Result](#) [PostSetValueRequest](#) ([AAX_CParamID](#) iParameterID, double iNormalizedValue) const [AAX_OVERRIDE](#)
- [AAX_Result](#) [PostCurrentValue](#) ([AAX_CParamID](#) iParameterID, double iNormalizedValue) const [AAX_OVERRIDE](#)
- [AAX_Result](#) [PostTouchRequest](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)
- [AAX_Result](#) [PostReleaseRequest](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)
- [AAX_Result](#) [GetTouchState](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *outTouched) [AAX_OVERRIDE](#)
- [AAX_Result](#) [ParameterNameChanged](#) ([AAX_CParamID](#) iParameterID) [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_IAutomationDelegate](#)

- virtual [~AAX_IAutomationDelegate](#) ()
- virtual [AAX_Result](#) [RegisterParameter](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result](#) [UnregisterParameter](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result](#) [PostSetValueRequest](#) ([AAX_CParamID](#) iParameterID, double normalizedValue) const =0
- virtual [AAX_Result](#) [PostCurrentValue](#) ([AAX_CParamID](#) iParameterID, double normalizedValue) const =0
- virtual [AAX_Result](#) [PostTouchRequest](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result](#) [PostReleaseRequest](#) ([AAX_CParamID](#) iParameterID)=0
- virtual [AAX_Result](#) [GetTouchState](#) ([AAX_CParamID](#) iParameterID, [AAX_CBoolean](#) *oTouched)=0
- virtual [AAX_Result](#) [ParameterNameChanged](#) ([AAX_CParamID](#) iParameterID)=0

14.146.2 Constructor & Destructor Documentation

14.146.2.1 AAX_VAutomationDelegate()

```
AAX_VAutomationDelegate::AAX_VAutomationDelegate (
    IACFUnknown * pUnknown )
```

14.146.2.2 ~AAX_VAutomationDelegate()

```
AAX_VAutomationDelegate::~~AAX_VAutomationDelegate ( )
```

14.146.3 Member Function Documentation**14.146.3.1 GetUnknown()**

```
IACFUnknown * AAX_VAutomationDelegate::GetUnknown ( ) const [inline]
```

14.146.3.2 RegisterParameter()

```
AAX_Result AAX_VAutomationDelegate::RegisterParameter (
    AAX_CParamID iParameterID ) [virtual]
```

Register a control with the automation system using a char* based control identifier

The automation delegate owns a list of the IDs of all of the parameters that have been registered with it. This list is used to set up listeners for all of the registered parameters such that the automation delegate may update the plug-in when the state of any of the registered parameters have been modified.

See also

[AAX_IAutomationDelegate::UnregisterParameter\(\)](#)

Parameters

in	<i>iParameterID</i>	Parameter ID that is being registered
----	---------------------	---------------------------------------

Implements [AAX_IAutomationDelegate](#).

14.146.3.3 UnregisterParameter()

```
AAX_Result AAX_VAutomationDelegate::UnregisterParameter (
    AAX_CParamID iParameterID ) [virtual]
```

Unregister a control with the automation system using a char* based control identifier

Note

All registered controls should be unregistered or the system might leak.

See also

[AAX_IAutomationDelegate::RegisterParameter\(\)](#)

Parameters

in	<i>iParameterID</i>	Parameter ID that is being registered
----	---------------------	---------------------------------------

Implements [AAX_IAutomationDelegate](#).

14.146.3.4 PostSetValueRequest()

```
AAX_Result AAX_VAutomationDelegate::PostSetValueRequest (
    AAX_CParamID iParameterID,
    double normalizedValue ) const [virtual]
```

Submits a request for the given parameter's value to be changed

Parameters

in	<i>iParameterID</i>	ID of the parameter for which a change is requested
in	<i>normalizedValue</i>	The requested new parameter value, formatted as a double and normalized to [0 1]

Implements [AAX_IAutomationDelegate](#).

14.146.3.5 PostCurrentValue()

```
AAX_Result AAX_VAutomationDelegate::PostCurrentValue (
    AAX_CParamID iParameterID,
    double normalizedValue ) const [virtual]
```

Notifies listeners that a parameter's value has changed

Parameters

in	<i>iParameterID</i>	ID of the parameter that has been updated
in	<i>normalizedValue</i>	The current parameter value, formatted as a double and normalized to [0 1]

Implements [AAX_IAutomationDelegate](#).

14.146.3.6 PostTouchRequest()

```
AAX_Result AAX_VAutomationDelegate::PostTouchRequest (
    AAX_CParamID iParameterID ) [virtual]
```

Requests that the given parameter be "touched", i.e. locked for updates by the current client

Parameters

in	<i>iParameterID</i>	ID of the parameter that will be touched
----	---------------------	--

Implements [AAX_IAutomationDelegate](#).

14.146.3.7 PostReleaseRequest()

```
AAX_Result AAX_VAutomationDelegate::PostReleaseRequest (
    AAX_CParamID iParameterID ) [virtual]
```

Requests that the given parameter be "released", i.e. available for updates from any client

Parameters

in	<i>iParameterID</i>	ID of the parameter that will be released
----	---------------------	---

Implements [AAX_IAutomationDelegate](#).

14.146.3.8 GetTouchState()

```
AAX_Result AAX_VAutomationDelegate::GetTouchState (
    AAX_CParamID iParameterID,
    AAX_CBoolean * oTouched ) [virtual]
```

Gets the current touched state of a parameter

Parameters

in	<i>iParameterID</i>	ID of the parameter that is being queried
out	<i>oTouched</i>	The current touch state of the parameter

Implements [AAX_IAutomationDelegate](#).

14.146.3.9 ParameterNameChanged()

```
AAX_Result AAX_VAutomationDelegate::ParameterNameChanged (
    AAX_CParamID iParameterID ) [virtual]
```

Notify listeners that the parameter's display name has changed

Note that this is not part of the underlying automation delegate interface with the host; it is converted on the AAX side to a notification posted to the host via the [AAX_IController](#) .

Parameters

in	<i>iParameterID</i>	ID of the parameter that has been updated
----	---------------------	---

Implements [AAX_IAutomationDelegate](#).

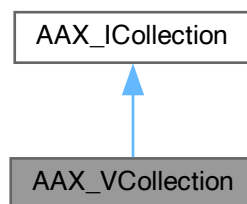
The documentation for this class was generated from the following file:

- [AAX_VAutomationDelegate.h](#)

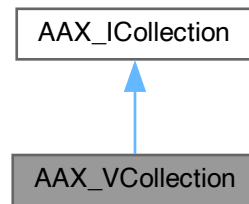
14.147 AAX_VCollection Class Reference

```
#include <AAX_VCollection.h>
```

Inheritance diagram for AAX_VCollection:



Collaboration diagram for AAX_VCollection:



14.147.1 Description

Version-managed concrete [AAX_ICollection](#) class.

Public Member Functions

- [AAX_VCollection](#) ([IACFUnknown](#) *pUnkHost)
- [~AAX_VCollection](#) () [AAX_OVERRIDE](#)
- [AAX_IEffectDescriptor](#) * [NewDescriptor](#) () [AAX_OVERRIDE](#)
Create a new Effect descriptor.
- [AAX_Result](#) [AddEffect](#) (const char *inEffectID, [AAX_IEffectDescriptor](#) *inEffectDescriptor) [AAX_OVERRIDE](#)
Add an Effect description to the collection.
- [AAX_Result](#) [SetManufacturerName](#) (const char *inPackageName) [AAX_OVERRIDE](#)
Set the plug-in manufacturer name.
- [AAX_Result](#) [AddPackageName](#) (const char *inPackageName) [AAX_OVERRIDE](#)
Set the plug-in package name.
- [AAX_Result](#) [SetPackageVersion](#) (uint32_t inVersion) [AAX_OVERRIDE](#)
Set the plug-in package version number.
- [AAX_IPropertyMap](#) * [NewPropertyMap](#) () [AAX_OVERRIDE](#)
Create a new property map.
- [AAX_Result](#) [SetProperties](#) ([AAX_IPropertyMap](#) *inProperties) [AAX_OVERRIDE](#)
Set the properties of the collection.
- [AAX_IDescriptionHost](#) * [DescriptionHost](#) () [AAX_OVERRIDE](#)
- const [AAX_IDescriptionHost](#) * [DescriptionHost](#) () const [AAX_OVERRIDE](#)
- [IACFDefinition](#) * [HostDefinition](#) () const [AAX_OVERRIDE](#)
- [IACFPluginDefinition](#) * [GetUnknown](#) (void) const

Public Member Functions inherited from [AAX_ICollection](#)

- virtual [~AAX_ICollection](#) ()
- virtual [AAX_IEffectDescriptor * NewDescriptor](#) ()=0
Create a new Effect descriptor.
- virtual [AAX_Result AddEffect](#) (const char *inEffectID, [AAX_IEffectDescriptor *inEffectDescriptor](#))=0
Add an Effect description to the collection.
- virtual [AAX_Result SetManufacturerName](#) (const char *inPackageName)=0
Set the plug-in manufacturer name.
- virtual [AAX_Result AddPackageName](#) (const char *inPackageName)=0
Set the plug-in package name.
- virtual [AAX_Result SetPackageVersion](#) (uint32_t inVersion)=0
Set the plug-in package version number.
- virtual [AAX_IPropertyMap * NewPropertyMap](#) ()=0
Create a new property map.
- virtual [AAX_Result SetProperties](#) ([AAX_IPropertyMap *inProperties](#))=0
Set the properties of the collection.
- virtual [AAX_IDescriptionHost * DescriptionHost](#) ()=0
- virtual const [AAX_IDescriptionHost * DescriptionHost](#) () const =0
- virtual [IACFDefinition * HostDefinition](#) () const =0

14.147.2 Constructor & Destructor Documentation

14.147.2.1 [AAX_VCollection](#)()

```
AAX_VCollection::AAX_VCollection (
    IACFUnknown \* pUnkHost )
```

14.147.2.2 [~AAX_VCollection](#)()

```
AAX_VCollection::~~AAX_VCollection ( )
```

14.147.3 Member Function Documentation

14.147.3.1 [NewDescriptor](#)()

```
AAX\_IEffectDescriptor \* AAX\_VCollection::NewDescriptor ( ) [virtual]
```

Create a new Effect descriptor.

This implementation retains each generated [AAX_IEffectDescriptor](#) and destroys the descriptor upon [AAX_VCollection](#) destruction

Create a new Effect descriptor.

Implements [AAX_ICollection](#).

14.147.3.2 AddEffect()

```
AAX_Result AAX_VCollection::AddEffect (
    const char * inEffectID,
    AAX_IEffectDescriptor * inEffectDescriptor ) [virtual]
```

Add an Effect description to the collection.

Each Effect that a plug-in registers with [AAX_ICollection::AddEffect\(\)](#) is considered a completely different user-facing product. For example, in Avid's Dynamics III plug-in the Expander, Compressor, and DeEsser are each registered as separate Effects. All stem format variations within each Effect are registered within that Effect's [AAX_IEffectDescriptor](#) using [AddComponent\(\)](#).

The [AAX_eProperty_ProductID](#) value for all ProcessProcs within a single Effect must be identical.

This method passes ownership of an [AAX_IEffectDescriptor](#) object to the [AAX_ICollection](#). The [AAX_IEffectDescriptor](#) must not be deleted by the AAX plug-in, nor should it be edited in any way after it is passed to the [AAX_ICollection](#).

Parameters

in	<i>inEffectID</i>	The effect ID.
in	<i>inEffectDescriptor</i>	The Effect descriptor.

Implements [AAX_ICollection](#).

14.147.3.3 SetManufacturerName()

```
AAX_Result AAX_VCollection::SetManufacturerName (
    const char * inPackageName ) [virtual]
```

Set the plug-in manufacturer name.

Parameters

in	<i>inPackageName</i>	The name of the manufacturer.
----	----------------------	-------------------------------

Implements [AAX_ICollection](#).

14.147.3.4 AddPackageName()

```
AAX_Result AAX_VCollection::AddPackageName (
    const char * inPackageName ) [virtual]
```

Set the plug-in package name.

May be called multiple times to add abbreviated package names.

Note

Every plug-in must include at least one name variant with 16 or fewer characters, plus a null terminating character. Used for Presets folder.

Parameters

in	<i>inPackageName</i>	The name of the package.
----	----------------------	--------------------------

Implements [AAX_ICollection](#).

14.147.3.5 SetPackageVersion()

```
AAX_Result AAX_VCollection::SetPackageVersion (
    uint32_t inVersion ) [virtual]
```

Set the plug-in package version number.

Parameters

in	<i>inVersion</i>	The package version number.
----	------------------	-----------------------------

Implements [AAX_ICollection](#).

14.147.3.6 NewPropertyMap()

```
AAX_IPropertyMap * AAX_VCollection::NewPropertyMap ( ) [virtual]
```

Create a new property map.

Implements [AAX_ICollection](#).

14.147.3.7 SetProperties()

```
AAX_Result AAX_VCollection::SetProperties (
    AAX_IPropertyMap * inProperties ) [virtual]
```

Set the properties of the collection.

Parameters

in	<i>inProperties</i>	Collection properties
----	---------------------	-----------------------

Implements [AAX_ICollection](#).

14.147.3.8 DescriptionHost() [1/2]

```
AAX_IDescriptionHost * AAX_VCollection::DescriptionHost ( ) [virtual]
```

Get a pointer to an [AAX_IDescriptionHost](#), if supported by the host

This interface is served by the [AAX_ICollection](#) in order to avoid requiring a new method prototype for the [GetEffectDescriptions\(\)](#) method called from the AAX Library.

See also

[AAX_UIDs.h](#) for available feature UUIDs, e.g. [AAXATTR_ClientFeature_AuxOutputStem](#)

Implements [AAX_ICollection](#).

14.147.3.9 DescriptionHost() [2/2]

```
const AAX_IDescriptionHost * AAX_VCollection::DescriptionHost ( ) const [virtual]
```

Get a pointer to an [AAX_IDescriptionHost](#), if supported by the host

This interface is served by the [AAX_ICollection](#) in order to avoid requiring a new method prototype for the [GetEffectDescriptions\(\)](#) method called from the AAX Library.

See also

[AAX_UIDs.h](#) for available feature UUIDs, e.g. [AAXATTR_ClientFeature_AuxOutputStem](#)

Implements [AAX_ICollection](#).

14.147.3.10 HostDefinition()

```
IACFDefinition * AAX_VCollection::HostDefinition ( ) const [virtual]
```

Get a pointer to an [IACFDefinition](#), if supported by the host

This interface is served by the [AAX_ICollection](#) in order to avoid requiring a new method prototype for the [GetEffectDescriptions\(\)](#) method called from the AAX Library.

See also

[AAX_UIDs.h](#) for available host attribute UUIDs, e.g. [AAXATTR_Client_Level](#)

The implementation of [AAX_ICollection](#) owns the referenced object. No AddRef occurs.

[IACFDefinition::DefineAttribute\(\)](#) is not supported on this object

Implements [AAX_ICollection](#).

14.147.3.11 GetUnknown()

```
IACFPluginDefinition * AAX_VCollection::GetUnknown (
    void ) const
```

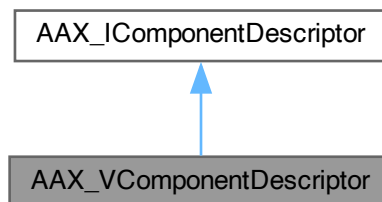
The documentation for this class was generated from the following file:

- [AAX_VCollection.h](#)

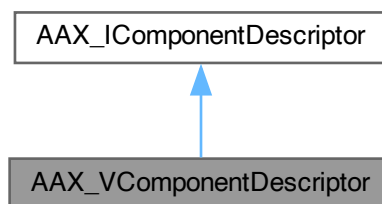
14.148 AAX_VComponentDescriptor Class Reference

```
#include <AAX_VComponentDescriptor.h>
```

Inheritance diagram for AAX_VComponentDescriptor:



Collaboration diagram for AAX_VComponentDescriptor:



14.148.1 Description

Version-managed concrete [AAX_IComponentDescriptor](#) class.

Public Member Functions

- [AAX_VComponentDescriptor](#) (IACFUnknown *pUnkHost)
- [~AAX_VComponentDescriptor](#) () [AAX_OVERRIDE](#)
- [AAX_Result Clear](#) () [AAX_OVERRIDE](#)
Clears the descriptor.
- [AAX_Result AddReservedField](#) (AAX_CFieldIndex inFieldIndex, uint32_t inFieldType) [AAX_OVERRIDE](#)
Subscribes a context field to host-provided services or information.
- [AAX_Result AddAudioIn](#) (AAX_CFieldIndex inFieldIndex) [AAX_OVERRIDE](#)
Subscribes an audio input context field.
- [AAX_Result AddAudioOut](#) (AAX_CFieldIndex inFieldIndex) [AAX_OVERRIDE](#)
Subscribes an audio output context field.
- [AAX_Result AddAudioBufferLength](#) (AAX_CFieldIndex inFieldIndex) [AAX_OVERRIDE](#)
Subscribes a buffer length context field.
- [AAX_Result AddSampleRate](#) (AAX_CFieldIndex inFieldIndex) [AAX_OVERRIDE](#)
Subscribes a sample rate context field.
- [AAX_Result AddClock](#) (AAX_CFieldIndex inFieldIndex) [AAX_OVERRIDE](#)
Subscribes a clock context field.
- [AAX_Result AddSideChainIn](#) (AAX_CFieldIndex inFieldIndex) [AAX_OVERRIDE](#)
Subscribes a side-chain input context field.
- [AAX_Result AddDataInPort](#) (AAX_CFieldIndex inFieldIndex, uint32_t inPacketSize, AAX_EDataInPortType inPortType) [AAX_OVERRIDE](#)
Adds a custom data port to the algorithm context.
- [AAX_Result AddAuxOutputStem](#) (AAX_CFieldIndex inFieldIndex, int32_t inStemFormat, const char inNameUTF8[]) [AAX_OVERRIDE](#)
Adds an auxiliary output stem for a plug-in.
- [AAX_Result AddPrivateData](#) (AAX_CFieldIndex inFieldIndex, int32_t inDataSize, uint32_t inOptions) [AAX_OVERRIDE](#)
Adds a private data port to the algorithm context.
- [AAX_Result AddTemporaryData](#) (AAX_CFieldIndex inFieldIndex, uint32_t inDataElementSize) [AAX_OVERRIDE](#)
Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.
- [AAX_Result AddDmaInstance](#) (AAX_CFieldIndex inFieldIndex, AAX_IDma::EMode inDmaMode) [AAX_OVERRIDE](#)
Adds a DMA field to the plug-in's context.
- [AAX_Result AddMeters](#) (AAX_CFieldIndex inFieldIndex, const AAX_CTypeID *inMeterIDs, const uint32_t inMeterCount) [AAX_OVERRIDE](#)
Adds a meter field to the plug-in's context.
- [AAX_Result AddMIDINode](#) (AAX_CFieldIndex inFieldIndex, AAX_EMIDINodeType inNodeType, const char inNodeName[], uint32_t channelMask) [AAX_OVERRIDE](#)
Adds a MIDI node field to the plug-in's context.
- [AAX_IPropertyMap * NewPropertyMap](#) () const [AAX_OVERRIDE](#)
Creates a new, empty property map.
- [AAX_IPropertyMap * DuplicatePropertyMap](#) (AAX_IPropertyMap *inPropertyMap) const [AAX_OVERRIDE](#)
Creates a new property map using an existing property map.
- virtual [AAX_Result AddProcessProc_Native](#) (AAX_CProcessProc inProcessProc, AAX_IPropertyMap *inProperties=NULL, AAX_CInstanceInitProc inInstanceInitProc=NULL, AAX_CBackgroundProc inBackgroundProc=NULL, AAX_CSelector *outProcID=NULL) [AAX_OVERRIDE](#)
Registers an algorithm processing endpoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc_TI](#) (const char inDLLFileNameUTF8[], const char inProcessProcSymbol[], AAX_IPropertyMap *inProperties=NULL, const char inInstanceInitProcSymbol[]=NULL, const char inBackgroundProcSymbol[]=NULL, AAX_CSelector *outProcID=NULL) [AAX_OVERRIDE](#)
Registers an algorithm processing endpoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc](#) (AAX_IPropertyMap *inProperties, AAX_CSelector *outProcIDs=NULL, int32_t inProcIDsSize=0) [AAX_OVERRIDE](#)
Registers one or more algorithm processing endpoints (process procedures)
- IACFUnknown * [GetIUnknown](#) (void) const

Public Member Functions inherited from [AAX_IComponentDescriptor](#)

- virtual [~AAX_IComponentDescriptor](#) ()
- virtual [AAX_Result Clear](#) ()=0
Clears the descriptor.
- virtual [AAX_Result AddAudioIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio input context field.
- virtual [AAX_Result AddAudioOut](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes an audio output context field.
- virtual [AAX_Result AddAudioBufferLength](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a buffer length context field.
- virtual [AAX_Result AddSampleRate](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a sample rate context field.
- virtual [AAX_Result AddClock](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a clock context field.
- virtual [AAX_Result AddSideChainIn](#) ([AAX_CFieldIndex](#) inFieldIndex)=0
Subscribes a side-chain input context field.
- virtual [AAX_Result AddDataInPort](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inPacketSize, [AAX_EDataInPortType](#) inPortType=[AAX_eDataInPortType_Buffered](#))=0
Adds a custom data port to the algorithm context.
- virtual [AAX_Result AddAuxOutputStem](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inStemFormat, [const char](#) inNameUTF8[])=0
Adds an auxiliary output stem for a plug-in.
- virtual [AAX_Result AddPrivateData](#) ([AAX_CFieldIndex](#) inFieldIndex, [int32_t](#) inDataSize, [uint32_t](#) inOptions=[AAX_ePrivateDataOptions_DefaultOptions](#))=0
Adds a private data port to the algorithm context.
- virtual [AAX_Result AddTemporaryData](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inDataElementSize)=0
Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.
- virtual [AAX_Result AddDmaInstance](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_IDma::EMode](#) inDmaMode)=0
Adds a DMA field to the plug-in's context.
- virtual [AAX_Result AddMeters](#) ([AAX_CFieldIndex](#) inFieldIndex, [const AAX_CTypeID](#) *inMeterIDs, [const uint32_t](#) inMeterCount)=0
Adds a meter field to the plug-in's context.
- virtual [AAX_Result AddMIDINode](#) ([AAX_CFieldIndex](#) inFieldIndex, [AAX_EMIDINodeType](#) inNodeType, [const char](#) inNodeName[], [uint32_t](#) channelMask)=0
Adds a MIDI node field to the plug-in's context.
- virtual [AAX_Result AddReservedField](#) ([AAX_CFieldIndex](#) inFieldIndex, [uint32_t](#) inFieldType)=0
Subscribes a context field to host-provided services or information.
- virtual [AAX_IPropertyMap * NewPropertyMap](#) () [const](#) =0
Creates a new, empty property map.
- virtual [AAX_IPropertyMap * DuplicatePropertyMap](#) ([AAX_IPropertyMap](#) *inPropertyMap) [const](#) =0
Creates a new property map using an existing property map.
- virtual [AAX_Result AddProcessProc_Native](#) ([AAX_CProcessProc](#) inProcessProc, [AAX_IPropertyMap](#) *inProperties=NULL, [AAX_CInstanceInitProc](#) inInstanceInitProc=NULL, [AAX_CBackgroundProc](#) inBackgroundProc=NULL, [AAX_CSelector](#) *outProcID=NULL)=0
Registers an algorithm processing entrypoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc_TI](#) ([const char](#) inDLLFileNameUTF8[], [const char](#) inProcessProcSymbol[], [AAX_IPropertyMap](#) *inProperties, [const char](#) inInstanceInitProcSymbol[]=NULL, [const char](#) inBackgroundProcSymbol[]=NULL, [AAX_CSelector](#) *outProcID=NULL)=0
Registers an algorithm processing entrypoint (process procedure) for the native architecture.
- virtual [AAX_Result AddProcessProc](#) ([AAX_IPropertyMap](#) *inProperties, [AAX_CSelector](#) *outProcIDs=NULL, [int32_t](#) inProcIDsSize=0)=0

Registers one or more algorithm processing entrypoints (process procedures)

- `template<typename aContextType >`
`AAX_Result AddProcessProc_Native` (void([AAX_CALLBACK](#) *inProcessProc)(aContextType *const inInstancesBegin[], const void *inInstancesEnd), [AAX_IPropertyMap](#) *inProperties=NULL, int32_t([AAX_CALLBACK](#) *inInstanceInitProc)(const aContextType *inInstanceContextPtr, [AAX_EComponentInstanceInitAction](#) inAction)=NULL, int32_t([AAX_CALLBACK](#) *inBackgroundProc)(void)=NULL)

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

Friends

- class [AAX_VPropertyMap](#)

14.148.2 Constructor & Destructor Documentation

14.148.2.1 AAX_VComponentDescriptor()

```
AAX_VComponentDescriptor::AAX_VComponentDescriptor (
    IACFUnknown * pUnkHost )
```

14.148.2.2 ~AAX_VComponentDescriptor()

```
AAX_VComponentDescriptor::~~AAX_VComponentDescriptor ( )
```

14.148.3 Member Function Documentation

14.148.3.1 Clear()

```
AAX\_Result AAX_VComponentDescriptor::Clear ( ) [virtual]
```

Clears the descriptor.

Clears the descriptor and readies it for the next algorithm description

Implements [AAX_IComponentDescriptor](#).

14.148.3.2 AddReservedField()

```
AAX\_Result AAX_VComponentDescriptor::AddReservedField (
    AAX\_CFieldIndex inFieldIndex,
    uint32_t inFieldType ) [virtual]
```

Subscribes a context field to host-provided services or information.

Note

Currently for internal use only.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inFieldType</i>	Type of field that is being added

Implements [AAX_IComponentDescriptor](#).

14.148.3.3 AddAudioIn()

```
AAX_Result AAX_VComponentDescriptor::AddAudioIn (
    AAX_CFieldIndex inFieldIndex ) [virtual]
```

Subscribes an audio input context field.

Defines an audio in port for host-provided information in the algorithm's context structure.

- Data type: float**
- Data kind: An array of float arrays, one for each input channel

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implements [AAX_IComponentDescriptor](#).

14.148.3.4 AddAudioOut()

```
AAX_Result AAX_VComponentDescriptor::AddAudioOut (
    AAX_CFieldIndex inFieldIndex ) [virtual]
```

Subscribes an audio output context field.

Defines an audio out port for host-provided information in the algorithm's context structure.

- Data type: float**
- Data kind: An array of float arrays, one for each output channel

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implements [AAX_IComponentDescriptor](#).

14.148.3.5 AddAudioBufferLength()

```
AAX_Result AAX_VComponentDescriptor::AddAudioBufferLength (
    AAX_CFieldIndex inFieldIndex ) [virtual]
```

Subscribes a buffer length context field.

Defines a buffer length port for host-provided information in the algorithm's context structure.

- Data type: `int32_t*`
- Data kind: The number of samples in the current audio buffer

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implements [AAX_IComponentDescriptor](#).

14.148.3.6 AddSampleRate()

```
AAX_Result AAX_VComponentDescriptor::AddSampleRate (
    AAX_CFieldIndex inFieldIndex ) [virtual]
```

Subscribes a sample rate context field.

Defines a sample rate port for host-provided information in the algorithm's context structure.

- Data type: [AAX_CSampleRate](#) *
- Data kind: The current sample rate

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implements [AAX_IComponentDescriptor](#).

14.148.3.7 AddClock()

```
AAX_Result AAX_VComponentDescriptor::AddClock (
```

```
AAX_CFieldIndex inFieldIndex ) [virtual]
```

Subscribes a clock context field.

Defines a clock port for host-provided information in the algorithm's context structure.

- Data type: [AAX_CTimestamp](#) *
- Data kind: A running counter which increments even when the transport is not playing. The counter increments exactly once per sample quantum.

Host Compatibility Notes As of Pro Tools 11.1, this field may be used in both Native and DSP plug-ins. The DSP clock data is a 16-bit cycling counter. This field was only available for Native plug-ins in previous Pro Tools versions.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implements [AAX_IComponentDescriptor](#).

14.148.3.8 AddSideChainIn()

```
AAX_Result AAX_VComponentDescriptor::AddSideChainIn (
    AAX_CFieldIndex inFieldIndex ) [virtual]
```

Subscribes a side-chain input context field.

Defines a side-chain input port for host-provided information in the algorithm's context structure.

- Data type: int32_t*
- Data kind: The index of the plug-in's first side-chain input channel within the array of input audio buffers

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
----	---------------------	--

Implements [AAX_IComponentDescriptor](#).

14.148.3.9 AddDataInPort()

```
AAX_Result AAX_VComponentDescriptor::AddDataInPort (
    AAX_CFieldIndex inFieldIndex,
```

```
uint32_t inPacketSize,
AAX_EDataInPortType inPortType ) [virtual]
```

Adds a custom data port to the algorithm context.

Defines a read-only data port for plug-in information in the algorithm's context structure. The plug-in can send information to this port using [AAX_IController::PostPacket\(\)](#).

The host guarantees that all packets will be delivered to this port in the order in which they were posted, up to the point of a packet buffer overflow, though some packets may be dropped depending on the `inPortType` and host implementation.

Note

When a plug-in is operating in offline (AudioSuite) mode, all data ports operate as [AAX_eDataInPortType_Unbuffered](#) ports

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inPacketSize</i>	Size of the data packets that will be sent to this port
in	<i>inPortType</i>	The requested packet delivery behavior for this port

Implements [AAX_IComponentDescriptor](#).

14.148.3.10 AddAuxOutputStem()

```
AAX_Result AAX_VComponentDescriptor::AddAuxOutputStem (
    AAX_CFieldIndex inFieldIndex,
    int32_t inStemFormat,
    const char inNameUTF8[] ) [virtual]
```

Adds an auxiliary output stem for a plug-in.

Use this method to add additional output channels to the algorithm context.

The aux output stem audio buffers will be added to the end of the audio outputs array in the order in which they are described. When writing audio data to a specific aux output, find the proper starting channel by accumulating all of the channels of the main output stem format and any previously-described aux output stems.

The plug-in is responsible for providing a meaningful name for each aux outputs. At the very least, individual outputs should be labeled "Output xx", where "xx" is the aux output number as it is defined in the plug-in. The output name should also include the words "mono" and "stereo" to support when users are looking for an output with a specific stem format.

Host Compatibility Notes There is a hard limit to the number of outputs that Pro Tools supports for a single plug-in instance. This limit is currently set at 256 channels, which includes all of the plug-in's output channels in addition to the sum total of all of its aux output stem channels.

Host Compatibility Notes Pro Tools supports only mono and stereo auxiliary output stem formats

Warning

This method will return an error code on hosts which do not support auxiliary output stems. This indicates that the host will not provide audio buffers for auxiliary output stems during processing. A plug-in must not attempt to write data into auxiliary output stem buffers which have not been provided by the host!

Parameters

in	<i>inFieldIndex</i>	DEPRECATED: This parameter is no longer needed by the host, but is included in the interface for binary compatibility
in	<i>inStemFormat</i>	The stem format of the new aux output
in	<i>inNameUTF8</i>	The name of the aux output. This name is static and cannot be changed after the descriptor is submitted to the host

Implements [AAX_IComponentDescriptor](#).

14.148.3.11 AddPrivateData()

```
AAX_Result AAX_VComponentDescriptor::AddPrivateData (
    AAX_CFieldIndex inFieldIndex,
    int32_t inDataSize,
    uint32_t inOptions ) [virtual]
```

Adds a private data port to the algorithm context.

Defines a read/write data port for private state data. Data written to this port will be maintained by the host between calls to the algorithm context.

See also

`alg_pd_registration`

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inDataSize</i>	Size of the data packets that will be sent to this port
in	<i>inOptions</i>	Options that define the private data port's behavior

Implements [AAX_IComponentDescriptor](#).

14.148.3.12 AddTemporaryData()

```
AAX_Result AAX_VComponentDescriptor::AddTemporaryData (
    AAX_CFieldIndex inFieldIndex,
    uint32_t inDataElementSize ) [virtual]
```

Adds a block of data to a context that is not saved between callbacks and is scaled by the system buffer size.

This can be very useful if you use block processing and need to store intermediate results. Just specify your base element size and the system will scale the overall block size by the buffer size. For example, to create a buffer of floats that is the length of the block, specify 4 bytes as the elementsize.

This data block does not retain state across callback and can also be reused across instances on memory constrained systems.

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the port, generated using AAX_FIELD_INDEX
in	<i>inDataElementSize</i>	The size of a single piece of data in the block. This number will be multiplied by the processing block size to determine total block size.

Implements [AAX_IComponentDescriptor](#).

14.148.3.13 AddDmaInstance()

```
AAX_Result AAX_VComponentDescriptor::AddDmaInstance (
    AAX_CFieldIndex inFieldIndex,
    AAX_IDma::EMode inDmaMode ) [virtual]
```

Adds a DMA field to the plug-in's context.

DMA (direct memory access) provides efficient reads from and writes to external memory on the DSP. DMA behavior is emulated in host-based plug-ins for cross-platform portability.

Note

The order in which DMA instances are added defines their priority and therefore order of execution of DMA operations. In most plug-ins, Scatter fields should be placed first in order to achieve the lowest possible access latency.

For more information, see [Direct Memory Access](#) .

Todo Update the DMA system management such that operation priority can be set arbitrarily

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inDmaMode</i>	AAX_IDma::EMode that will apply to this field

Implements [AAX_IComponentDescriptor](#).

14.148.3.14 AddMeters()

```
AAX_Result AAX_VComponentDescriptor::AddMeters (
    AAX_CFieldIndex inFieldIndex,
    const AAX_CTypeID * inMeterIDs,
    const uint32_t inMeterCount ) [virtual]
```

Adds a meter field to the plug-in's context.

Meter fields include an array of meter tap values, with one tap per meter per context. Only one meter field should be added per Component. Individual meter behaviors can be described at the Effect level.

For more information, see [Plug-in meters](#) .

Parameters

in	<i>inFieldIndex</i>	Unique identifier for the field, generated using AAX_FIELD_INDEX
in	<i>inMeterIDs</i>	Array of 32-bit IDs, one for each meter. Meter IDs must be unique within the Effect.
in	<i>inMeterCount</i>	The number of meters included in this field

Implements [AAX_IComponentDescriptor](#).

14.148.3.15 AddMIDINode()

```
AAX_Result AAX_VComponentDescriptor::AddMIDINode (
    AAX_CFieldIndex inFieldIndex,
    AAX_EMIDINodeType inNodeType,
    const char inNodeName[],
    uint32_t channelMask ) [virtual]
```

Adds a MIDI node field to the plug-in's context.

- Data type: [AAX_IMIDINode](#) *

The resulting MIDI node data will be available both in the algorithm context and in the plug-in's [data model](#) via [UpdateMIDINodes\(\)](#).

To add a MIDI node that is only accessible to the plug-in's data model, use [AAX_IEffectDescriptor::AddControlMIDINode\(\)](#)

Host Compatibility Notes Due to current restrictions MIDI data won't be delivered to DSP algorithms, only to AAX Native.

Parameters

in	<i>inFieldIndex</i>	The ID of the port. MIDI node ports should formatted as a pointer to an AAX_IMIDINode .
in	<i>inNodeType</i>	The type of MIDI node, as AAX_EMIDINodeType
in	<i>inNodeName</i>	The name of the MIDI node as it should appear in the host's UI
in	<i>channelMask</i>	The channel mask for the MIDI node. This parameter specifies used MIDI channels. For Global MIDI nodes, use a mask of AAX_EMidiGlobalNodeSelectors

Implements [AAX_IComponentDescriptor](#).

14.148.3.16 NewPropertyMap()

```
AAX_IPropertyMap * AAX_VComponentDescriptor::NewPropertyMap ( ) const [virtual]
```

Creates a new, empty property map.

The component descriptor owns the reference to the resulting property map, and the underlying property map is destroyed when the component descriptor is released.

This implementation retains each generated [AAX_IPropertyMap](#) and destroys the property map upon [AAX_VComponentDescriptor](#) destruction

Implements [AAX_IComponentDescriptor](#).

14.148.3.17 DuplicatePropertyMap()

```
AAX_IPropertyMap * AAX_VComponentDescriptor::DuplicatePropertyMap (
    AAX_IPropertyMap * inPropertyMap ) const [virtual]
```

Creates a new property map using an existing property map.

The component descriptor owns the reference to the resulting property map, and the underlying property map is destroyed when the component descriptor is released.

Parameters

in	<i>inPropertyMap</i>	The property values in this map will be copied into the new map
----	----------------------	---

This implementation retains each generated [AAX_IPropertyMap](#) and destroys the property map upon [AAX_VComponentDescriptor](#) destruction

Implements [AAX_IComponentDescriptor](#).

14.148.3.18 AddProcessProc_Native()

```
virtual AAX_Result AAX_VComponentDescriptor::AddProcessProc_Native (
    AAX_CProcessProc inProcessProc,
    AAX_IPropertyMap * inProperties = NULL,
    AAX_CInstanceInitProc inInstanceInitProc = NULL,
    AAX_CBackgroundProc inBackgroundProc = NULL,
    AAX_CSelector * outProcID = NULL ) [virtual]
```

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

Parameters

in	<i>inProcessProc</i>	Symbol for this processing callback
in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
in	<i>inInstanceInitProc</i>	Initialization routine that will be called when a new instance of the Effect is created. See Algorithm initialization .
in	<i>inBackgroundProc</i>	Background routine that will be called in an idle context within the same address space as the associated process procedure. See Background processing callback
Generated by Doxygen out	<i>outProcID</i>	

Todo document this parameter

Implements [AAX_IComponentDescriptor](#).

14.148.3.19 AddProcessProc_TI()

```
virtual AAX_Result AAX_VComponentDescriptor::AddProcessProc_TI (
    const char inDLLFileNameUTF8[],
    const char inProcessProcSymbol[],
    AAX_IPropertyMap * inProperties = NULL,
    const char inInstanceInitProcSymbol[] = NULL,
    const char inBackgroundProcSymbol[] = NULL,
    AAX_CSelector * outProcID = NULL ) [virtual]
```

Registers an algorithm processing entrypoint (process procedure) for the native architecture.

Parameters

in	<i>inDLLFileNameUTF8</i>	UTF-8 encoded filename for the ELF DLL containing the algorithm code fragment
in	<i>inProcessProcSymbol</i>	Symbol for this processing callback
in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new ProcessProc. The property map contents are unchanged and the map may be re-used when registering additional ProcessProcs.
in	<i>inInstanceInitProcSymbol</i>	Initialization routine that will be called when a new instance of the Effect is created. Must be included in the same DLL as the main algorithm entrypoint. See Algorithm initialization .
in	<i>inBackgroundProcSymbol</i>	Background routine that will be called in an idle context within the same address space as the associated process procedure. Must be included in the same DLL as the main algorithm entrypoint. See Background processing callback
out	<i>outProcID</i>	

Todo document this parameter

Implements [AAX_IComponentDescriptor](#).

14.148.3.20 AddProcessProc()

```
virtual AAX_Result AAX_VComponentDescriptor::AddProcessProc (
    AAX_IPropertyMap * inProperties,
    AAX_CSelector * outProcIDs = NULL,
    int32_t inProcIDsSize = 0 ) [virtual]
```

Registers one or more algorithm processing entrypoints (process procedures)

Any non-overlapping set of processing entrypoints may be specified. Typically this can be used to specify both Native and TI entrypoints using the same call.

The AAX Library implementation of this method includes backwards compatibility logic to complete the `ProcessProc` registration on hosts which do not support this method. Therefore plug-in code may use this single registration routine instead of separate calls to `AddProcessProc_Native()`, `AddProcessProc_TI()`, etc. regardless of the host version.

The following properties replace the input arguments to the platform-specific registration methods:

`AddProcessProc_Native()` (`AAX_eProperty_PluginID_Native`, `AAX_eProperty_PluginID_AudioSuite`)

- `AAX_CProcessProc` `iProcessProc`: `AAX_eProperty_NativeProcessProc` (required)
- `AAX_CInstanceInitProc` `iInstanceInitProc`: `AAX_eProperty_NativeInstanceInitProc` (optional)
- `AAX_CBackgroundProc` `iBackgroundProc`: `AAX_eProperty_NativeBackgroundProc` (optional)

`AddProcessProc_TI()` (`AAX_eProperty_PluginID_TI`)

- `const char` `inDLLFileNameUTF8[]`: `AAX_eProperty_TIDLLFileName` (required)
- `const char` `iProcessProcSymbol[]`: `AAX_eProperty_TIPProcessProc` (required)
- `const char` `iInstanceInitProcSymbol[]`: `AAX_eProperty_TIInstanceInitProc` (optional)
- `const char` `iBackgroundProcSymbol[]`: `AAX_eProperty_TIBackgroundProc` (optional)

If any platform-specific plug-in ID property is present in `iProperties` then `AddProcessProc()` will check for the required properties for that platform.

Note

`AAX_eProperty_AudioBufferLength` will be ignored for the Native and AudioSuite ProcessProcs since it should only be used for AAX DSP.

Parameters

in	<i>inProperties</i>	A property map for this processing callback. The property map's values are copied by the host and associated with the new <code>ProcessProc</code> . The property map contents are unchanged and the map may be re-used when registering additional <code>ProcessProcs</code> .
out	<i>outProcIDs</i>	

Todo document this parameter Returned array will be NULL-terminated

Parameters

in	<i>inProcIDsSize</i>	The size of the array provided to <code>oProcIDs</code> . If <code>oProcIDs</code> is non-NULL but <code>iProcIDsSize</code> is not large enough for all of the registered <code>ProcessProcs</code> (plus one for NULL termination) then this method will fail with <code>AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW</code>
----	----------------------	--

Implements [AAX_IComponentDescriptor](#).

14.148.3.21 GetUnknown()

```
IACFUnknown * AAX_VComponentDescriptor::GetUnknown (
    void ) const
```

14.148.4 Friends And Related Function Documentation

14.148.4.1 AAX_VPropertyMap

```
friend class AAX\_VPropertyMap [friend]
```

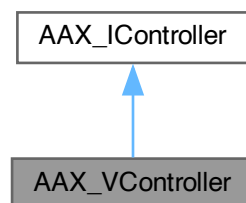
The documentation for this class was generated from the following file:

- [AAX_VComponentDescriptor.h](#)

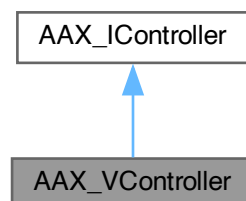
14.149 AAX_VController Class Reference

```
#include <AAX_VController.h>
```

Inheritance diagram for AAX_VController:



Collaboration diagram for AAX_VController:



14.149.1 Description

Version-managed concrete [Controller](#) class.

For usage information, see [Host-provided interfaces](#)

Public Member Functions

- [AAX_VController](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VController](#) () override
- [AAX_Result](#) [GetEffectID](#) ([AAX_IString](#) *outEffectID) const [AAX_OVERRIDE](#)
- [AAX_Result](#) [GetSampleRate](#) ([AAX_CSampleRate](#) *outSampleRate) const [AAX_OVERRIDE](#)
CALL: Returns the current literal sample rate.
- [AAX_Result](#) [GetInputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const [AAX_OVERRIDE](#)
CALL: Returns the plug-in's input stem format.
- [AAX_Result](#) [GetOutputStemFormat](#) ([AAX_EStemFormat](#) *outStemFormat) const [AAX_OVERRIDE](#)
CALL: Returns the plug-in's output stem format.
- [AAX_Result](#) [GetSignalLatency](#) ([int32_t](#) *outSamples) const [AAX_OVERRIDE](#)
CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.
- [AAX_Result](#) [GetHybridSignalLatency](#) ([int32_t](#) *outSamples) const [AAX_OVERRIDE](#)
CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.
- [AAX_Result](#) [GetPlugInTargetPlatform](#) ([AAX_CTargetPlatform](#) *outTargetPlatform) const [AAX_OVERRIDE](#)
CALL: Returns execution platform type, native or TI.
- [AAX_Result](#) [GetIsAudioSuite](#) ([AAX_CBoolean](#) *outIsAudioSuite) const [AAX_OVERRIDE](#)
CALL: Returns true for AudioSuite instances.
- [AAX_Result](#) [GetCycleCount](#) ([AAX_EProperty](#) inWhichCycleCount, [AAX_CPropertyValue](#) *outNumCycles) const [AAX_OVERRIDE](#)
CALL: returns the plug-in's current real-time DSP cycle count.
- [AAX_Result](#) [GetTODLocation](#) ([AAX_CTimeOfDay](#) *outTODLocation) const [AAX_OVERRIDE](#)
CALL: Returns the current Time Of Day (TOD) of the system.
- [AAX_Result](#) [GetCurrentAutomationTimestamp](#) ([AAX_CTransportCounter](#) *outTimestamp) const [AAX_OVERRIDE](#)
CALL: Returns the current automation timestamp if called during the [GenerateCoefficients\(\)](#) call AND the generation of coefficients is being triggered by an automation point instead of immediate changes.
- [AAX_Result](#) [GetHostName](#) ([AAX_IString](#) *outHostNameString) const [AAX_OVERRIDE](#)
CALL: Returns name of the host application this plug-in instance is being loaded by. This string also typically includes version information.
- [AAX_Result](#) [SetSignalLatency](#) ([int32_t](#) inNumSamples) [AAX_OVERRIDE](#)
CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.
- [AAX_Result](#) [SetCycleCount](#) ([AAX_EProperty](#) *inWhichCycleCounts, [AAX_CPropertyValue](#) *iValues, [int32_t](#) inNumValues) [AAX_OVERRIDE](#)
CALL: Indicates a change in the plug-in's real-time DSP cycle count.
- [AAX_Result](#) [PostPacket](#) ([AAX_CFieldIndex](#) inFieldIndex, const void *inPayloadP, [uint32_t](#) inPayloadSize) [AAX_OVERRIDE](#)
CALL: Posts a data packet to the host for routing between plug-in components.
- [AAX_Result](#) [SendNotification](#) ([AAX_CTypeID](#) inNotificationType, const void *inNotificationData, [uint32_t](#) inNotificationDataSize) [AAX_OVERRIDE](#)
CALL: Dispatch a notification.
- [AAX_Result](#) [SendNotification](#) ([AAX_CTypeID](#) inNotificationType) [AAX_OVERRIDE](#)
CALL: Sends an event to the GUI (no payload)

*Note**Not an AAX interface method*

- [AAX_Result GetCurrentMeterValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterValue) const [AAX_OVERRIDE](#)
CALL: Retrieves the current value of a host-managed plug-in meter.
- [AAX_Result GetMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID, float *outMeterPeakValue) const [AAX_OVERRIDE](#)
CALL: Retrieves the currently held peak value of a host-managed plug-in meter.
- [AAX_Result ClearMeterPeakValue](#) ([AAX_CTypeID](#) inMeterID) const [AAX_OVERRIDE](#)
CALL: Clears the peak value from a host-managed plug-in meter.
- [AAX_Result GetMeterClipped](#) ([AAX_CTypeID](#) inMeterID, [AAX_CBoolean](#) *outClipped) const [AAX_OVERRIDE](#)
CALL: Retrieves the clipped flag from a host-managed plug-in meter.
- [AAX_Result ClearMeterClipped](#) ([AAX_CTypeID](#) inMeterID) const [AAX_OVERRIDE](#)
CALL: Clears the clipped flag from a host-managed plug-in meter.
- [AAX_Result GetMeterCount](#) (uint32_t *outMeterCount) const [AAX_OVERRIDE](#)
CALL: Retrieves the number of host-managed meters registered by a plug-in.
- [AAX_Result GetNextMIDIPacket](#) ([AAX_CFieldIndex](#) *outPort, [AAX_CMidiPacket](#) *outPacket) [AAX_OVERRIDE](#)
CALL: Retrieves MIDI packets for described MIDI nodes.
- [AAX_IPageTable * CreateTableCopyForEffect](#) ([AAX_CPropertyValue](#) inManufacturerID, [AAX_CPropertyValue](#) inProductID, [AAX_CPropertyValue](#) inPlugInID, uint32_t inTableType, int32_t inTablePageSize) const [AAX_OVERRIDE](#)
Copy the current page table data for a particular plug-in type.
- [AAX_IPageTable * CreateTableCopyForLayout](#) (const char *inEffectID, const char *inLayoutName, uint32_t inTableType, int32_t inTablePageSize) const [AAX_OVERRIDE](#)
Copy the current page table data for a particular plug-in effect and page table layout.
- [AAX_IPageTable * CreateTableCopyForEffectFromFile](#) (const char *inPageTableFilePath, [AAX_ETextEncoding](#) inFilePathEncoding, [AAX_CPropertyValue](#) inManufacturerID, [AAX_CPropertyValue](#) inProductID, [AAX_CPropertyValue](#) inPlugInID, uint32_t inTableType, int32_t inTablePageSize) const [AAX_OVERRIDE](#)
Copy the current page table data for a particular plug-in type.
- [AAX_IPageTable * CreateTableCopyForLayoutFromFile](#) (const char *inPageTableFilePath, [AAX_ETextEncoding](#) inFilePathEncoding, const char *inLayoutName, uint32_t inTableType, int32_t inTablePageSize) const [AAX_OVERRIDE](#)
Copy the current page table data for a particular plug-in effect and page table layout.

Public Member Functions inherited from [AAX_IController](#)

- virtual [~AAX_IController](#) (void)

14.149.2 Constructor & Destructor Documentation**14.149.2.1 [AAX_VController\(\)](#)**

```
AAX_VController::AAX_VController (
    IACFUnknown * pUnknown )
```

14.149.2.2 [~AAX_VController\(\)](#)

```
AAX_VController::~~AAX_VController ( ) [override]
```

14.149.3 Member Function Documentation

14.149.3.1 GetEffectID()

```
AAX_Result AAX_VController::GetEffectID (
    AAX_IString * outEffectID ) const [virtual]
```

Implements [AAX_IController](#).

14.149.3.2 GetSampleRate()

```
AAX_Result AAX_VController::GetSampleRate (
    AAX_CSampleRate * outSampleRate ) const [virtual]
```

CALL: Returns the current literal sample rate.

Parameters

out	<i>outSampleRate</i>	The current sample rate
-----	----------------------	-------------------------

Implements [AAX_IController](#).

14.149.3.3 GetInputStemFormat()

```
AAX_Result AAX_VController::GetInputStemFormat (
    AAX_EStemFormat * outStemFormat ) const [virtual]
```

CALL: Returns the plug-in's input stem format.

Parameters

out	<i>outStemFormat</i>	The current input stem format
-----	----------------------	-------------------------------

Implements [AAX_IController](#).

14.149.3.4 GetOutputStemFormat()

```
AAX_Result AAX_VController::GetOutputStemFormat (
    AAX_EStemFormat * outStemFormat ) const [virtual]
```

CALL: Returns the plug-in's output stem format.

Parameters

out	<i>outStemFormat</i>	The current output stem format
-----	----------------------	--------------------------------

Implements [AAX_IController](#).

14.149.3.5 GetSignalLatency()

```
AAX_Result AAX_VController::GetSignalLatency (
    int32_t * outSamples ) const [virtual]
```

CALL: Returns the most recent signal (algorithmic) latency that has been published by the plug-in.

This method provides the most recently published signal latency. The host may not have updated its delay compensation to match this signal latency yet, so plug-ins that dynamically change their latency using [SetSignalLatency\(\)](#) should always wait for an [AAX_eNotificationEvent_SignalLatencyChanged](#) notification before updating its algorithm to incur this latency.

See also

[SetSignalLatency\(\)](#)

Parameters

out	<i>outSamples</i>	The number of samples of signal delay published by the plug-in
-----	-------------------	--

Implements [AAX_IController](#).

14.149.3.6 GetHybridSignalLatency()

```
AAX_Result AAX_VController::GetHybridSignalLatency (
    int32_t * outSamples ) const [virtual]
```

CALL: Returns the latency between the algorithm normal input samples and the inputs returning from the hybrid component.

This method provides the number of samples that the AAX host expects the plug-in to delay a signal. The host will use this value when accounting for latency across the system.

Note

This value will generally scale up with sample rate, although it's not a simple multiple due to some fixed overhead. This value will be fixed for any given sample rate regardless of other buffer size settings in the host app.

Parameters

out	<i>outSamples</i>	The number of samples of hybrid signal delay
-----	-------------------	--

Implements [AAX_IController](#).

14.149.3.7 GetPlugInTargetPlatform()

```
AAX_Result AAX_VController::GetPlugInTargetPlatform (
    AAX_CTargetPlatform * outTargetPlatform ) const [virtual]
```

CALL: Returns execution platform type, native or TI.

Parameters

out	<i>outTargetPlatform</i>	The type of the current execution platform as one of AAX_ETargetPlatform .
-----	--------------------------	--

Implements [AAX_IController](#).

14.149.3.8 GetIsAudioSuite()

```
AAX_Result AAX_VController::GetIsAudioSuite (
    AAX_CBoolean * outIsAudioSuite ) const [virtual]
```

CALL: Returns true for AudioSuite instances.

Parameters

out	<i>outIsAudioSuite</i>	The boolean flag which indicate true for AudioSuite instances.
-----	------------------------	--

Implements [AAX_IController](#).

14.149.3.9 GetCycleCount()

```
AAX_Result AAX_VController::GetCycleCount (
    AAX_EProperty inWhichCycleCount,
    AAX_CPropertyValue * outNumCycles ) const [virtual]
```

CALL: returns the plug-in's current real-time DSP cycle count.

This method provides the number of cycles that the AAX host expects the DSP plug-in to consume. The host uses this value when allocating DSP resources for the plug-in.

Note

A plug-in should never apply a DSP algorithm with more demanding resource requirements than what is currently accounted for by the host. To set a higher cycle count value, a plug-in must call [AAX_IController::SetCycleCount\(\)](#), then poll [AAX_IController::GetCycleCount\(\)](#) until the new value has been applied. Once the host has recognized the new cycle count value, the plug-in may apply the more demanding algorithm.

Parameters

in	<i>inWhichCycleCount</i>	Selector for the requested cycle count metric. One of: <ul style="list-style-type: none"> • AAX_eProperty_TI_SharedCycleCount • AAX_eProperty_TI_InstanceCycleCount • AAX_eProperty_TI_MaxInstancesPerChip
in	<i>outNumCycles</i>	The current value of the selected cycle count metric

Todo PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

Implements [AAX_IController](#).

14.149.3.10 GetTODLocation()

```
AAX_Result AAX_VController::GetTODLocation (
    AAX_CTimeOfDay * outTODLocation ) const [virtual]
```

CALL: Returns the current Time Of Day (TOD) of the system.

This method provides a plug-in the TOD (in samples) of the current system. TOD is the number of samples that the playhead has traversed since the beginning of playback.

Note

The TOD value is the immediate value of the audio engine playhead. This value is incremented within the audio engine's real-time rendering context; it is not synchronized with non-real-time calls to plug-in interface methods.

Parameters

out	<i>outTODLocation</i>	The current Time Of Day as set by the host
-----	-----------------------	--

Implements [AAX_IController](#).

14.149.3.11 GetCurrentAutomationTimestamp()

```
AAX_Result AAX_VController::GetCurrentAutomationTimestamp (
    AAX_CTransportCounter * outTimestamp ) const [virtual]
```

CALL: Returns the current automation timestamp if called during the [GenerateCoefficients\(\)](#) call AND the generation of coefficients is being triggered by an automation point instead of immediate changes.

Note

This function will return 0 if called from outside of [GenerateCoefficients\(\)](#) or if the [GenerateCoefficients\(\)](#) call was initiated due to a non-automated change. In those cases, you can get your sample offset from the transport start using [GetTODLocation\(\)](#).

Parameters

out	<i>outTimestamp</i>	The current coefficient timestamp. Sample count from transport start.
-----	---------------------	---

Implements [AAX_IController](#).

14.149.3.12 GetHostName()

```
AAX_Result AAX_VController::GetHostName (
    AAX_IString * outHostNameString ) const [virtual]
```

CALL: Returns name of the host application this plug-in instance is being loaded by. This string also typically includes version information.

Host Compatibility Notes Pro Tools versions from Pro Tools 11.0 to Pro Tools 12.3.1 will return a generic version string to this call. This issue is resolved beginning in Pro Tools 12.4.

Parameters

out	<i>outHostNameString</i>	The name of the current host application.
-----	--------------------------	---

Implements [AAX_IController](#).

14.149.3.13 SetSignalLatency()

```
AAX_Result AAX_VController::SetSignalLatency (
    int32_t inNumSamples ) [virtual]
```

CALL: Submits a request to change the delay compensation value that the host uses to account for the plug-in's signal (algorithmic) latency.

This method is used to request a change in the number of samples that the AAX host expects the plug-in to delay a signal.

The host is not guaranteed to immediately apply the new latency value. A plug-in should avoid incurring an actual algorithmic latency that is different than the latency accounted for by the host.

To set a new latency value, a plug-in must call [AAX_IController::SetSignalLatency\(\)](#), then wait for an [AAX_eNotificationEvent_SignalLatencyChanged](#) notification. Once this notification has been received, [AAX_IController::GetSignalLatency\(\)](#) will reflect the updated latency value and the plug-in should immediately apply any relevant algorithmic changes that alter its latency to this new value.

Warning

Parameters which affect the latency of a plug-in should not be made available for control through automation. This will result in audible glitches when delay compensation is adjusted while playing back automation for these parameters.

Parameters

<i>in</i>	<i>inNumSamples</i>	The number of samples of signal delay that the plug-in requests to incur
-----------	---------------------	--

Implements [AAX_IController](#).

14.149.3.14 SetCycleCount()

```
AAX_Result AAX_VController::SetCycleCount (
    AAX_EProperty * inWhichCycleCounts,
    AAX_CPropertyValue * iValues,
    int32_t numValues ) [virtual]
```

CALL: Indicates a change in the plug-in's real-time DSP cycle count.

This method is used to request a change in the number of cycles that the AAX host expects the DSP plug-in to consume.

Note

A plug-in should never apply a DSP algorithm with more demanding resource requirements than what is currently accounted for by the host. To set a higher cycle count value, a plug-in must call [AAX_IController::SetCycleCount\(\)](#), then poll [AAX_IController::GetCycleCount\(\)](#) until the new value has been applied. Once the host has recognized the new cycle count value, the plug-in may apply the more demanding algorithm.

Parameters

in	<i>inWhichCycleCounts</i>	Array of selectors indicating the specific cycle count metrics that should be set. Each selector must be one of: <ul style="list-style-type: none"> • AAX_eProperty_TI_SharedCycleCount • AAX_eProperty_TI_InstanceCycleCount • AAX_eProperty_TI_MaxInstancesPerChip
in	<i>iValues</i>	An array of values requested, one for each of the selected cycle count metrics.
in	<i>numValues</i>	The size of <i>iValues</i>

Todo PLACEHOLDER - NOT CURRENTLY IMPLEMENTED IN HOST

Implements [AAX_IController](#).

14.149.3.15 PostPacket()

```
AAX_Result AAX_VController::PostPacket (
    AAX_CFieldIndex inFieldIndex,
    const void * inPayloadP,
    uint32_t inPayloadSize ) [virtual]
```

CALL: Posts a data packet to the host for routing between plug-in components.

The posted packet is identified with a [AAX_CFieldIndex](#) packet index value, which is equivalent to the target data port's identifier. The packet's payload must have the expected size for the given packet index / data port, as defined when the port is created in [Describe](#). See [AAX_IComponentDescriptor::AddDataInPort\(\)](#).

Warning

Any data structures that will be passed between platforms (for example, sent to a TI DSP in an AAX DSP plug-in) must be properly data-aligned for compatibility across both platforms. See [AAX_ALIGN_FILE_ALG](#) for more information about guaranteeing cross-platform compatibility of data structures used for algorithm processing.

Note

All calls to this method should be made within the scope of [AAX_IEffectParameters::GenerateCoefficients\(\)](#). Calls from outside this method may result in packets not being delivered. See [PT-206161](#)

Parameters

in	<i>inFieldIndex</i>	The packet's destination port
in	<i>inPayloadP</i>	A pointer to the packet's payload data
in	<i>inPayloadSize</i>	The size, in bytes, of the payload data

Implements [AAX_IController](#).

14.149.3.16 SendNotification() [1/2]

```
AAX_Result AAX_VController::SendNotification (
    AAX_CTypeID inNotificationType,
    const void * inNotificationData,
    uint32_t inNotificationDataSize ) [virtual]
```

CALL: Dispatch a notification.

The notification is handled by the host and may be delivered back to other plug-in components such as the GUI or data model (via [AAX_IEffectGUI::NotificationReceived\(\)](#) or [AAX_IEffectParameters::NotificationReceived\(\)](#), respectively) depending on the notification type.

The host may choose to dispatch the posted notification either synchronously or asynchronously.

See the [AAX_ENotificationEvent](#) documentation for more information.

This method is supported by AAX V2 Hosts only. Check the return code on the return of this function. If the error is [AAX_ERROR_UNIMPLEMENTED](#), your plug-in is being loaded into a host that doesn't support this feature.

Parameters

in	<i>inNotificationType</i>	Type of notification to send
in	<i>inNotificationData</i>	Block of notification data
in	<i>inNotificationDataSize</i>	Size of <i>inNotificationData</i> , in bytes

Implements [AAX_IController](#).

14.149.3.17 SendNotification() [2/2]

```
AAX_Result AAX_VController::SendNotification (
    AAX_CTypeID inNotificationType ) [virtual]
```

CALL: Sends an event to the GUI (no payload)

Note

Not an [AAX](#) interface method

This version of the notification method is a convenience for notifications which do not take any payload data. Internally, it simply calls [AAX_IController::SendNotification\(AAX_CTypeID, const void*, uint32_t\)](#) with a null payload.

Parameters

in	<i>inNotificationType</i>	Type of notification to send
----	---------------------------	------------------------------

Note

Not an [AAX](#) interface method

Implements [AAX_IController](#).

14.149.3.18 GetCurrentMeterValue()

```
AAX_Result AAX_VController::GetCurrentMeterValue (
    AAX_CTypeID inMeterID,
    float * outMeterValue ) const [virtual]
```

CALL: Retrieves the current value of a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried
out	<i>outMeterValue</i>	The queried meter's current value

Implements [AAX_IController](#).

14.149.3.19 GetMeterPeakValue()

```
AAX_Result AAX_VController::GetMeterPeakValue (
    AAX_CTypeID inMeterID,
    float * outMeterPeakValue ) const [virtual]
```

CALL: Retrieves the currently held peak value of a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried
out	<i>outMeterPeakValue</i>	The queried meter's currently held peak value

Implements [AAX_IController](#).

14.149.3.20 ClearMeterPeakValue()

```
AAX_Result AAX_VController::ClearMeterPeakValue (
    AAX_CTypeID inMeterID ) const [virtual]
```

CALL: Clears the peak value from a host-managed plug-in meter.

Parameters

in	<i>inMeterID</i>	ID of the meter that is being cleared
----	------------------	---------------------------------------

Implements [AAX_IController](#).

14.149.3.21 GetMeterClipped()

```
AAX_Result AAX_VController::GetMeterClipped (
    AAX_CTypeID inMeterID,
    AAX_CBoolean * outClipped ) const [virtual]
```

CALL: Retrieves the clipped flag from a host-managed plug-in meter.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

in	<i>inMeterID</i>	ID of the meter that is being queried.
out	<i>outClipped</i>	The queried meter's clipped flag.

Implements [AAX_IController](#).

14.149.3.22 ClearMeterClipped()

```
AAX_Result AAX_VController::ClearMeterClipped (
    AAX_CTypeID inMeterID ) const [virtual]
```

CALL: Clears the clipped flag from a host-managed plug-in meter.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

in	<i>inMeterID</i>	ID of the meter that is being cleared.
----	------------------	--

Implements [AAX_IController](#).

14.149.3.23 GetMeterCount()

```
AAX_Result AAX_VController::GetMeterCount (
    uint32_t * outMeterCount ) const [virtual]
```

CALL: Retrieves the number of host-managed meters registered by a plug-in.

See [AAX_IComponentDescriptor::AddMeters\(\)](#).

Parameters

out	<i>outMeterCount</i>	The number of registered plug-in meters.
-----	----------------------	--

Implements [AAX_IController](#).

14.149.3.24 GetNextMIDIpacket()

```
AAX_Result AAX_VController::GetNextMIDIpacket (
    AAX_CFieldIndex * outPort,
    AAX_CMidiPacket * outPacket ) [virtual]
```

CALL: Retrieves MIDI packets for described MIDI nodes.

Parameters

out	<i>outPort</i>	port ID of the MIDI node that has unhandled packet
out	<i>outPacket</i>	The MIDI packet

Implements [AAX_IController](#).

14.149.3.25 CreateTableCopyForEffect()

```
AAX_IPageTable * AAX_VController::CreateTableCopyForEffect (
    AAX_CPropertyValue inManufacturerID,
    AAX_CPropertyValue inProductID,
    AAX_CPropertyValue inPlugInID,
    uint32_t inTableType,
    int32_t inTablePageSize ) const [virtual]
```

Copy the current page table data for a particular plug-in type.

The host may restrict plug-ins to only copying page table data from certain plug-in types, such as plug-ins from the same manufacturer or plug-in types within the same effect.

See [Page Table Guide](#) for more information about page tables.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if the requested plug-in type is unknown, if *inTableType* is unknown or if *inTablePageSize* is not a supported size for the given table type.

Parameters

in	<i>inManufacturerID</i>	Manufacturer ID of the desired plug-in type
in	<i>inProductID</i>	Product ID of the desired plug-in type
in	<i>inPlugInID</i>	Type ID of the desired plug-in type (AAX_eProperty_PlugInID_Native , AAX_eProperty_PlugInID_TI)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implements [AAX_IController](#).

14.149.3.26 CreateTableCopyForLayout()

```
AAX_IPageTable * AAX_VController::CreateTableCopyForLayout (
    const char * inEffectID,
    const char * inLayoutName,
    uint32_t inTableType,
    int32_t inTablePageSize ) const [virtual]
```

Copy the current page table data for a particular plug-in effect and page table layout.

The host may restrict plug-ins to only copying page table data from certain effects, such as effects registered within the current [AAX](#) plug-in bundle.

See [Page Table Guide](#) for more information about page tables.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if the requested effect ID is unknown or if *inLayoutName* is not a valid layout name for the page tables registered for the effect.

Parameters

in	<i>inEffectID</i>	Effect ID for the desired effect. See AAX_ICollection::AddEffect()
in	<i>inLayoutName</i>	Page table layout name ("name" attribute of the <code>PTLayout</code> XML tag)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implements [AAX_IController](#).

14.149.3.27 CreateTableCopyForEffectFromFile()

```
AAX_IPageTable * AAX_VController::CreateTableCopyForEffectFromFile (
    const char * inPageTableFilePath,
```

```

AAX_ETextEncoding inFilePathEncoding,
AAX_CPropertyValue inManufacturerID,
AAX_CPropertyValue inProductID,
AAX_CPropertyValue inPlugInID,
uint32_t inTableType,
int32_t inTablePageSize ) const [virtual]

```

Copy the current page table data for a particular plug-in type.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if the requested plug-in type is unknown, if `inTableType` is unknown or if `inTablePageSize` is not a supported size for the given table type.

Parameters

in	<i>inPageTableFilePath</i>	Path to XML page table file.
in	<i>inFilePathEncoding</i>	File path text encoding.
in	<i>inManufacturerID</i>	Manufacturer ID of the desired plug-in type
in	<i>inProductID</i>	Product ID of the desired plug-in type
in	<i>inPlugInID</i>	Type ID of the desired plug-in type (AAX_eProperty_PlugInID_Native , AAX_eProperty_PlugInID_TI)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implements [AAX_IController](#).

14.149.3.28 CreateTableCopyForLayoutFromFile()

```

AAX_IPageTable * AAX_VController::CreateTableCopyForLayoutFromFile (
    const char * inPageTableFilePath,
    AAX_ETextEncoding inFilePathEncoding,
    const char * inLayoutName,
    uint32_t inTableType,
    int32_t inTablePageSize ) const [virtual]

```

Copy the current page table data for a particular plug-in effect and page table layout.

Returns

A new page table object to which the requested page table data has been copied. Ownership of this object passes to the caller.

a null pointer if `inLayoutName` is not a valid layout name for the page tables file.

Parameters

in	<i>inPageTableFilePath</i>	Path to XML page table file.
in	<i>inFilePathEncoding</i>	File path text encoding.
in	<i>inLayoutName</i>	Page table layout name ("name" attribute of the <code>PTLayout</code> XML tag)
in	<i>inTableType</i>	Four-char type identifier for the requested table type (e.g. 'PgTL', 'Av81', etc.)
in	<i>inTablePageSize</i>	Page size for the requested table. Some tables support multiple page sizes.

Implements [AAX_IController](#).

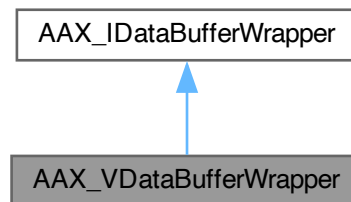
The documentation for this class was generated from the following file:

- [AAX_VController.h](#)

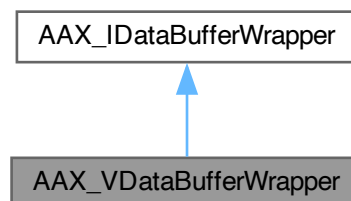
14.150 AAX_VDataBufferWrapper Class Reference

```
#include <AAX_VDataBufferWrapper.h>
```

Inheritance diagram for AAX_VDataBufferWrapper:



Collaboration diagram for AAX_VDataBufferWrapper:



14.150.1 Description

Wrapper for an [AAX_IDataBuffer](#).

Like [AAX_IController](#) and similar classes, this class provides a non-ACF interface matching an ACF interface, in this case [AAX_IACFDataBuffer](#) .

The implementation of this interface will contain a reference counted pointer to the underlying ACF interface. This interface may be extended with convenience functions that are not required on the underlying ACF interface.

Public Member Functions

- [AAX_VDataBufferWrapper](#) ([IACFUnknown](#) *iUnknown)
- [~AAX_VDataBufferWrapper](#) () [AAX_OVERRIDE](#)
- [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const [AAX_OVERRIDE](#)
- [AAX_Result Size](#) ([int32_t](#) *oSize) const [AAX_OVERRIDE](#)
- [AAX_Result Data](#) (void const **oBuffer) const [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_IDataBufferWrapper](#)

- virtual [~AAX_IDataBufferWrapper](#) ()=default
- virtual [AAX_Result Type](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_Result Size](#) ([int32_t](#) *oSize) const =0
- virtual [AAX_Result Data](#) (void const **oBuffer) const =0

14.150.2 Constructor & Destructor Documentation

14.150.2.1 [AAX_VDataBufferWrapper](#)()

```
AAX_VDataBufferWrapper::AAX_VDataBufferWrapper (
    IACFUnknown * iUnknown ) [explicit]
```

14.150.2.2 [~AAX_VDataBufferWrapper](#)()

```
AAX_VDataBufferWrapper::~~AAX_VDataBufferWrapper ( )
```

14.150.3 Member Function Documentation

14.150.3.1 Type()

```
AAX_Result AAX_VDataBufferWrapper::Type (
    AAX_CTypeID * oType ) const [virtual]
```

The type of data contained in this buffer

This identifier must be sufficient for a client that knows the type to correctly interpret and use the data.

Implements [AAX_IDataBufferWrapper](#).

14.150.3.2 Size()

```
AAX_Result AAX_VDataBufferWrapper::Size (
    int32_t * oSize ) const [virtual]
```

The number of bytes of data in this buffer

Implements [AAX_IDataBufferWrapper](#).

14.150.3.3 Data()

```
AAX_Result AAX_VDataBufferWrapper::Data (
    void const ** oBuffer ) const [virtual]
```

The buffer of data

Implements [AAX_IDataBufferWrapper](#).

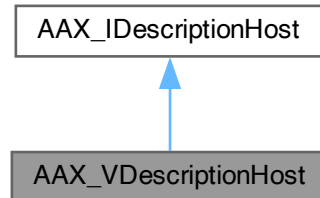
The documentation for this class was generated from the following file:

- [AAX_VDataBufferWrapper.h](#)

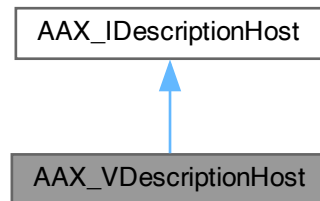
14.151 AAX_VDescriptionHost Class Reference

```
#include <AAX_VDescriptionHost.h>
```

Inheritance diagram for AAX_VDescriptionHost:



Collaboration diagram for AAX_VDescriptionHost:



14.151.1 Description

Versioned wrapper for access to host service interfaces provided during plug-in description

This object aggregates access to [AAX_IACFDescriptionHost](#) and [IACFDefinition](#), with support depending on the interface support level of the [IACFUnknown](#) which is passed to this object upon creation.

Public Member Functions

- [AAX_VDescriptionHost](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VDescriptionHost](#) () [AAX_OVERRIDE](#)
- const [AAX_IFeatureInfo](#) * [AcquireFeatureProperties](#) (const [AAX_Feature_UID](#) &inFeatureID) const [AAX_OVERRIDE](#)
- bool [Supported](#) () const
- [AAX_IACFDescriptionHost](#) * [DescriptionHost](#) ()
- const [AAX_IACFDescriptionHost](#) * [DescriptionHost](#) () const
- [IACFDefinition](#) * [HostDefinition](#) () const

Public Member Functions inherited from [AAX_IDescriptionHost](#)

- virtual [~AAX_IDescriptionHost](#) ()
- virtual const [AAX_IFeatureInfo](#) * [AcquireFeatureProperties](#) (const [AAX_Feature_UID](#) &inFeatureID) const =0

14.151.2 Constructor & Destructor Documentation**14.151.2.1 AAX_VDescriptionHost()**

```
AAX_VDescriptionHost::AAX_VDescriptionHost (
    IACFUnknown * pUnknown ) [explicit]
```

14.151.2.2 ~AAX_VDescriptionHost()

```
AAX_VDescriptionHost::~~AAX_VDescriptionHost ( )
```

14.151.3 Member Function Documentation**14.151.3.1 AcquireFeatureProperties()**

```
const AAX\_IFeatureInfo * AAX_VDescriptionHost::AcquireFeatureProperties (
    const AAX\_Feature\_UID & inFeatureID ) const [virtual]
```

Get the client's feature object for a given feature ID

Similar to [QueryInterface\(\)](#) but uses a feature identifier rather than a true IID

Ownership of the returned object is passed to the caller; the caller is responsible for destroying the object, e.g. by capturing the returned object in a smart pointer.

```
// AAX_IDescriptionHost* descHost
std::unique_ptr<const AAX\_IFeatureInfo> featureInfoPtr(descHost->AcquireFeatureProperties(someFeatureUID);
```

Returns

An [AAX_IFeatureInfo](#) interface with access to the host's feature properties for this feature.

NULL if the desired feature was not found or if an error occurred

Note

May return an [AAX_IFeatureInfo](#) object with limited method support, which would return an error such as [AAX_ERROR_NULL_OBJECT](#) or [AAX_ERROR_UNIMPLEMENTED](#) to interface calls.

If no [AAX_IFeatureInfo](#) is provided then that may mean that the host is unaware of the feature, or it may mean that the host is aware of the feature but has not implemented the AAX feature support interface for this feature yet.

Parameters

in	<i>inFeatureID</i>	Identifier of the requested feature
----	--------------------	-------------------------------------

Implements [AAX_IDescriptionHost](#).

14.151.3.2 Supported()

```
bool AAX_VDescriptionHost::Supported ( ) const [inline]
```

14.151.3.3 DescriptionHost() [1/2]

```
AAX\_IACFDescriptionHost * AAX_VDescriptionHost::DescriptionHost ( ) [inline]
```

14.151.3.4 DescriptionHost() [2/2]

```
const AAX\_IACFDescriptionHost * AAX_VDescriptionHost::DescriptionHost ( ) const [inline]
```

14.151.3.5 HostDefinition()

```
IACFDefinition * AAX_VDescriptionHost::HostDefinition ( ) const [inline]
```

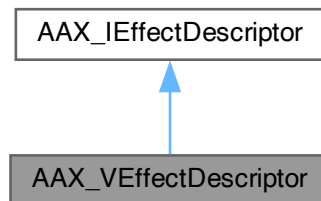
The documentation for this class was generated from the following file:

- [AAX_VDescriptionHost.h](#)

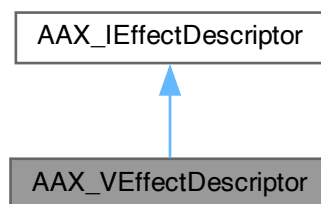
14.152 AAX_VEffectDescriptor Class Reference

```
#include <AAX_VEffectDescriptor.h>
```

Inheritance diagram for AAX_VEffectDescriptor:



Collaboration diagram for AAX_VEffectDescriptor:



14.152.1 Description

Version-managed concrete [AAX_IEffectDescriptor](#) class.

Public Member Functions

- [AAX_VEffectDescriptor](#) ([IACFUnknown](#) *pUnkHost)
- [~AAX_VEffectDescriptor](#) () [AAX_OVERRIDE](#)
- [AAX_IComponentDescriptor](#) * [NewComponentDescriptor](#) () [AAX_OVERRIDE](#)
Create an instance of a component descriptor.
- [AAX_Result](#) [AddComponent](#) ([AAX_IComponentDescriptor](#) *inComponentDescriptor) [AAX_OVERRIDE](#)
Add a component to an instance of a component descriptor.
- [AAX_Result](#) [AddName](#) (const char *inPlugInName) [AAX_OVERRIDE](#)
Add a name to the Effect.

- [AAX_Result AddCategory](#) (uint32_t inCategory) [AAX_OVERRIDE](#)
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- [AAX_Result AddCategoryBypassParameter](#) (uint32_t inCategory, [AAX_CParamID](#) inParamID) [AAX_OVERRIDE](#)
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- [AAX_Result AddProcPtr](#) (void *inProcPtr, [AAX_CProcPtrID](#) inProcID) [AAX_OVERRIDE](#)
Add a process pointer.
- [AAX_IPropertyMap * NewPropertyMap](#) () [AAX_OVERRIDE](#)
Create a new property map.
- [AAX_Result SetProperties](#) ([AAX_IPropertyMap](#) *inProperties) [AAX_OVERRIDE](#)
Set the properties of a new property map.
- [AAX_Result AddResourceInfo](#) ([AAX_EResourceType](#) inResourceType, const char *inInfo) [AAX_OVERRIDE](#)
Set resource file info.
- [AAX_Result AddMeterDescription](#) ([AAX_CTypeID](#) inMeterID, const char *inMeterName, [AAX_IPropertyMap](#) *inProperties) [AAX_OVERRIDE](#)
Add name and property map to meter with given ID.
- [AAX_Result AddControlMIDINode](#) ([AAX_CTypeID](#) inNodeID, [AAX_EMIDINodeType](#) inNodeType, const char inNodeName[], uint32_t inChannelMask) [AAX_OVERRIDE](#)
Add a control MIDI node to the plug-in data model.
- [IACFUnknown * GetIUnknown](#) (void) const

Public Member Functions inherited from [AAX_IEffectDescriptor](#)

- virtual [~AAX_IEffectDescriptor](#) ()
- virtual [AAX_IComponentDescriptor * NewComponentDescriptor](#) ()=0
Create an instance of a component descriptor.
- virtual [AAX_Result AddComponent](#) ([AAX_IComponentDescriptor](#) *inComponentDescriptor)=0
Add a component to an instance of a component descriptor.
- virtual [AAX_Result AddName](#) (const char *inPlugInName)=0
Add a name to the Effect.
- virtual [AAX_Result AddCategory](#) (uint32_t inCategory)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result AddCategoryBypassParameter](#) (uint32_t inCategory, [AAX_CParamID](#) inParamID)=0
Add a category to your plug-in. See [AAX_EPlugInCategory](#).
- virtual [AAX_Result AddProcPtr](#) (void *inProcPtr, [AAX_CProcPtrID](#) inProcID)=0
Add a process pointer.
- virtual [AAX_IPropertyMap * NewPropertyMap](#) ()=0
Create a new property map.
- virtual [AAX_Result SetProperties](#) ([AAX_IPropertyMap](#) *inProperties)=0
Set the properties of a new property map.
- virtual [AAX_Result AddResourceInfo](#) ([AAX_EResourceType](#) inResourceType, const char *inInfo)=0
Set resource file info.
- virtual [AAX_Result AddMeterDescription](#) ([AAX_CTypeID](#) inMeterID, const char *inMeterName, [AAX_IPropertyMap](#) *inProperties)=0
Add name and property map to meter with given ID.
- virtual [AAX_Result AddControlMIDINode](#) ([AAX_CTypeID](#) inNodeID, [AAX_EMIDINodeType](#) inNodeType, const char inNodeName[], uint32_t inChannelMask)=0
Add a control MIDI node to the plug-in data model.

14.152.2 Constructor & Destructor Documentation

14.152.2.1 AAX_VEffectDescriptor()

```
AAX_VEffectDescriptor::AAX_VEffectDescriptor (
    IACFUnknown * pUnkHost )
```

14.152.2.2 ~AAX_VEffectDescriptor()

```
AAX_VEffectDescriptor::~~AAX_VEffectDescriptor ( )
```

14.152.3 Member Function Documentation**14.152.3.1 NewComponentDescriptor()**

```
AAX_IComponentDescriptor * AAX_VEffectDescriptor::NewComponentDescriptor ( ) [virtual]
```

Create an instance of a component descriptor.

This implementation retains each generated [AAX_IComponentDescriptor](#) and destroys the property map upon [AAX_VEffectDescriptor](#) destruction

Implements [AAX_IEffectDescriptor](#).

14.152.3.2 AddComponent()

```
AAX_Result AAX_VEffectDescriptor::AddComponent (
    AAX_IComponentDescriptor * inComponentDescriptor ) [virtual]
```

Add a component to an instance of a component descriptor.

Unlike with [AAX_ICollection::AddEffect\(\)](#), the [AAX_IEffectDescriptor](#) does not take ownership of the [AAX_IComponentDescriptor](#) that is passed to it in this method. The host copies out the contents of this descriptor, and thus the plug-in may re-use the same descriptor object when creating additional similar components.

Parameters

in	<i>inComponentDescriptor</i>	
----	------------------------------	--

Implements [AAX_IEffectDescriptor](#).

14.152.3.3 AddName()

```
AAX_Result AAX_VEffectDescriptor::AddName (
    const char * inPlugInName ) [virtual]
```

Add a name to the Effect.

May be called multiple times to add abbreviated Effect names.

Note

Every Effect must include at least one name variant with 31 or fewer characters, plus a null terminating character

Parameters

in	<i>inPlugInName</i>	The name assigned to the plug-in.
----	---------------------	-----------------------------------

Implements [AAX_IEffectDescriptor](#).

14.152.3.4 AddCategory()

```
AAX_Result AAX_VEffectDescriptor::AddCategory (
    uint32_t inCategory ) [virtual]
```

Add a category to your plug-in. See [AAX_EPlugInCategory](#).

Parameters

in	<i>inCategory</i>	One of the categories for the plug-in.
----	-------------------	--

Implements [AAX_IEffectDescriptor](#).

14.152.3.5 AddCategoryBypassParameter()

```
AAX_Result AAX_VEffectDescriptor::AddCategoryBypassParameter (
    uint32_t inCategory,
    AAX_CParamID inParamID ) [virtual]
```

Add a category to your plug-in. See [AAX_EPlugInCategory](#).

Parameters

in	<i>inCategory</i>	One of the categories for the plug-in.
in	<i>inParamID</i>	The parameter ID of the parameter used to bypass the category section of the plug-in.

Implements [AAX_IEffectDescriptor](#).

14.152.3.6 AddProcPtr()

```
AAX_Result AAX_VEffectDescriptor::AddProcPtr (
    void * inProcPtr,
    AAX_CProcPtrID inProcID ) [virtual]
```

Add a process pointer.

Parameters

in	<i>inProcPtr</i>	A process pointer.
in	<i>inProcID</i>	A process ID.

Implements [AAX_IEffectDescriptor](#).

14.152.3.7 NewPropertyMap()

```
AAX_IPropertyMap * AAX_VEffectDescriptor::NewPropertyMap ( ) [virtual]
```

Create a new property map.

This implementation retains each generated [AAX_IPropertyMap](#) and destroys the property map upon [AAX_VEffectDescriptor](#) destruction

Implements [AAX_IEffectDescriptor](#).

14.152.3.8 SetProperties()

```
AAX_Result AAX_VEffectDescriptor::SetProperties (
    AAX_IPropertyMap * inProperties ) [virtual]
```

Set the properties of a new property map.

Parameters

in	<i>inProperties</i>	Description
----	---------------------	-------------

Implements [AAX_IEffectDescriptor](#).

14.152.3.9 AddResourceInfo()

```
AAX_Result AAX_VEffectDescriptor::AddResourceInfo (
    AAX_EResourceType inResourceType,
    const char * inInfo ) [virtual]
```

Set resource file info.

Parameters

in	<i>inResourceType</i>	See AAX_EResourceType.
in	<i>inInfo</i>	Definition varies on the resource type.

Implements [AAX_IEffectDescriptor](#).

14.152.3.10 AddMeterDescription()

```
AAX_Result AAX_VEffectDescriptor::AddMeterDescription (
    AAX_CTypeID inMeterID,
    const char * inMeterName,
    AAX_IPropertyMap * inProperties ) [virtual]
```

Add name and property map to meter with given ID.

Parameters

in	<i>inMeterID</i>	The ID of the meter being described.
in	<i>inMeterName</i>	The name of the meter.
in	<i>inProperties</i>	The property map containing meter related data such as meter type, orientation, etc.

Implements [AAX_IEffectDescriptor](#).

14.152.3.11 AddControlMIDINode()

```
AAX_Result AAX_VEffectDescriptor::AddControlMIDINode (
    AAX_CTypeID inNodeID,
    AAX_EMIDINodeType inNodeType,
    const char inNodeName[],
    uint32_t inChannelMask ) [virtual]
```

Add a control MIDI node to the plug-in data model.

- This MIDI node may receive note data as well as control data.
- To send MIDI data to the plug-in's algorithm, use [AAX_IComponentDescriptor::AddMIDINode\(\)](#).

See also

[AAX_IACFEffectorParameters_V2::UpdateControlMIDINodes\(\)](#)

Parameters

in	<i>inNodeID</i>	The ID for the new control MIDI node.
in	<i>inNodeType</i>	The type of the node.
in	<i>inNodeName</i>	The name of the node.
in	<i>inChannelMask</i>	The bit mask for required nodes channels (up to 16) or required global events for global node.

Implements [AAX_IEffectDescriptor](#).

14.152.3.12 GetUnknown()

```
IACFUnknown * AAX_VEffectDescriptor::GetUnknown (
    void ) const
```

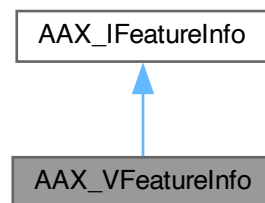
The documentation for this class was generated from the following file:

- [AAX_VEffectDescriptor.h](#)

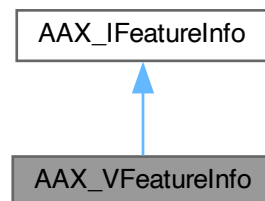
14.153 AAX_VFeatureInfo Class Reference

```
#include <AAX_VFeatureInfo.h>
```

Inheritance diagram for AAX_VFeatureInfo:



Collaboration diagram for AAX_VFeatureInfo:



14.153.1 Description

Concrete implementation of [AAX_IFeatureInfo](#), which provides a version-controlled interface to host feature information

Public Member Functions

- [AAX_VFeatureInfo](#) ([IACFUnknown](#) *pUnknown, const [AAX_Feature_UID](#) &inFeatureID)
- [~AAX_VFeatureInfo](#) () [AAX_OVERRIDE](#)
- [AAX_Result](#) [SupportLevel](#) ([AAX_ESupportLevel](#) &oSupportLevel) const [AAX_OVERRIDE](#)
- const [AAX_IPropertyMap](#) * [AcquireProperties](#) () const [AAX_OVERRIDE](#)
- const [AAX_Feature_UID](#) & [ID](#) () const [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_IFeatureInfo](#)

- virtual [~AAX_IFeatureInfo](#) ()
- virtual [AAX_Result](#) [SupportLevel](#) ([AAX_ESupportLevel](#) &oSupportLevel) const =0
- virtual const [AAX_IPropertyMap](#) * [AcquireProperties](#) () const =0
- virtual const [AAX_Feature_UID](#) & [ID](#) () const =0

14.153.2 Constructor & Destructor Documentation

14.153.2.1 [AAX_VFeatureInfo](#)()

```
AAX_VFeatureInfo::AAX_VFeatureInfo (
    IACFUnknown * pUnknown,
    const AAX\_Feature\_UID & inFeatureID ) [explicit]
```

14.153.2.2 [~AAX_VFeatureInfo](#)()

```
AAX_VFeatureInfo::~~AAX_VFeatureInfo ( )
```

14.153.3 Member Function Documentation

14.153.3.1 SupportLevel()

```
AAX_Result AAX_VFeatureInfo::SupportLevel (
    AAX_ESupportLevel & oSupportLevel ) const [virtual]
```

Determine the level of support for this feature by the host

Note

The host will not provide an underlying [AAX_IACFFeatureInfo](#) interface for features which it does not recognize at all, resulting in a [AAX_ERROR_NULL_OBJECT](#) error code

Implements [AAX_IFeatureInfo](#).

14.153.3.2 AcquireProperties()

```
const AAX_IPropertyMap * AAX_VFeatureInfo::AcquireProperties ( ) const [virtual]
```

Additional properties providing details of the feature support

See the feature's UID for documentation of which features provide additional properties

Ownership of the returned object is passed to the caller; the caller is responsible for destroying the object, e.g. by capturing the returned object in a smart pointer.

```
// AAX_IFeatureInfo* featureInfo
std::unique_ptr<const AAX_IPropertyMap> featurePropertiesPtr(featureInfo->AcquireProperties());
```

Returns

An [AAX_IPropertyMap](#) interface with access to the host's properties for this feature.

NULL if the desired feature was not found or if an error occurred

Note

May return an [AAX_IPropertyMap](#) object with limited method support, which would return an error such as [AAX_ERROR_NULL_OBJECT](#) or [AAX_ERROR_UNIMPLEMENTED](#) to interface calls.

Implements [AAX_IFeatureInfo](#).

14.153.3.3 ID()

```
const AAX_Feature_UID & AAX_VFeatureInfo::ID ( ) const [virtual]
```

Returns the ID of the feature which this object represents

Implements [AAX_IFeatureInfo](#).

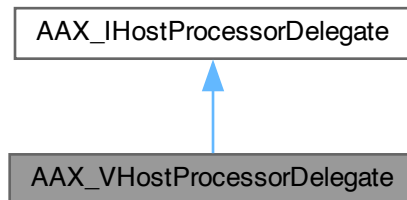
The documentation for this class was generated from the following file:

- [AAX_VFeatureInfo.h](#)

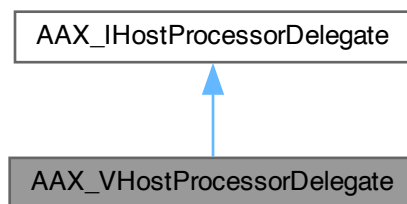
14.154 AAX_VHostProcessorDelegate Class Reference

```
#include <AAX_VHostProcessorDelegate.h>
```

Inheritance diagram for AAX_VHostProcessorDelegate:



Collaboration diagram for AAX_VHostProcessorDelegate:



14.154.1 Description

Version-managed concrete [Host Processor delegate](#) class.

Public Member Functions

- [AAX_VHostProcessorDelegate](#) ([IACFUnknown](#) *pUnknown)
- [AAX_Result GetAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int64_t inLocation, int32_t *ioNumSamples) [AAX_OVERRIDE](#)
CALL: Randomly access audio from the timeline.
- int32_t [GetSideChainInputNum](#) () [AAX_OVERRIDE](#)
CALL: Returns the index of the side chain input buffer.
- [AAX_Result ForceAnalyze](#) () [AAX_OVERRIDE](#)
CALL: Request an analysis pass.
- [AAX_Result ForceProcess](#) () [AAX_OVERRIDE](#)
CALL: Request a process pass.

Public Member Functions inherited from [AAX_IHostProcessorDelegate](#)

- virtual [~AAX_IHostProcessorDelegate](#) ()
- virtual [AAX_Result GetAudio](#) (const float *const inAudioIns[], int32_t inAudioInCount, int64_t inLocation, int32_t *ioNumSamples)=0
CALL: Randomly access audio from the timeline.
- virtual int32_t [GetSideChainInputNum](#) ()=0
CALL: Returns the index of the side chain input buffer.
- virtual [AAX_Result ForceAnalyze](#) ()=0
CALL: Request an analysis pass.
- virtual [AAX_Result ForceProcess](#) ()=0
CALL: Request a process pass.

14.154.2 Constructor & Destructor Documentation**14.154.2.1 AAX_VHostProcessorDelegate()**

```
AAX_VHostProcessorDelegate::AAX_VHostProcessorDelegate (
    IACFUnknown * pUnknown )
```

14.154.3 Member Function Documentation**14.154.3.1 GetAudio()**

```
AAX\_Result AAX_VHostProcessorDelegate::GetAudio (
    const float *const inAudioIns[],
    int32_t inAudioInCount,
    int64_t inLocation,
    int32_t * ioNumSamples ) [virtual]
```

CALL: Randomly access audio from the timeline.

Called from within [AAX_IHostProcessor::RenderAudio\(\)](#), this method fills a buffer of samples with randomly-accessed data from the current input processing region on the timeline, including any extra samples such as processing "handles".

Note

Plug-ins that use this feature must set [AAX_eProperty_UsesRandomAccess](#) to `true`

It is not possible to retrieve samples from outside of the current input processing region

Always check the return value of this method before using the randomly-accessed samples

Parameters

in	<i>inAudioIns</i>	Timeline audio buffer(s). This must be set to <code>inAudioIns</code> from AAX_IHostProcessor::RenderAudio()
in	<i>inAudioInCount</i>	Number of buffers in <code>inAudioIns</code> . This must be set to <code>inAudioInCount</code> from AAX_IHostProcessor::RenderAudio()
in	<i>inLocation</i>	A sample location relative to the beginning of the currently processed region, e.g. a value of 0 corresponds to the timeline location returned by AAX_CHostProcessor::GetSrcStart()
in, out	<i>ioNumSamples</i>	<ul style="list-style-type: none"> • Input: The maximum number of samples to read. • Output: The actual number of samples that were read from the timeline

Implements [AAX_IHostProcessorDelegate](#).

14.154.3.2 GetSideChainInputNum()

```
int32_t AAX_VHostProcessorDelegate::GetSideChainInputNum ( ) [virtual]
```

CALL: Returns the index of the side chain input buffer.

Called from within [AAX_IHostProcessor::RenderAudio\(\)](#), this method returns the index of the side chain input sample buffer within `inAudioIns`.

Implements [AAX_IHostProcessorDelegate](#).

14.154.3.3 ForceAnalyze()

```
AAX_Result AAX_VHostProcessorDelegate::ForceAnalyze ( ) [virtual]
```

CALL: Request an analysis pass.

Call this method to request an analysis pass from within the plug-in. Most plug-ins should rely on the host to trigger analysis passes when appropriate. However, plug-ins that require an analysis pass a) outside of the context of host-driven render or analysis, or b) when internal plug-in data changes may need to call [ForceAnalyze\(\)](#).

Implements [AAX_IHostProcessorDelegate](#).

14.154.3.4 ForceProcess()

```
AAX_Result AAX_VHostProcessorDelegate::ForceProcess ( ) [virtual]
```

CALL: Request a process pass.

Call this method to request a process pass from within the plug-in. If [AAX_eProperty_RequiresAnalysis](#) is defined, the resulting process pass will be preceded by an analysis pass. This method should only be used in rare circumstances by plug-ins that must launch processing outside of the normal host AudioSuite workflow.

Implements [AAX_IHostProcessorDelegate](#).

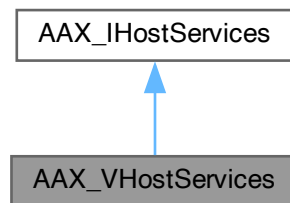
The documentation for this class was generated from the following file:

- [AAX_VHostProcessorDelegate.h](#)

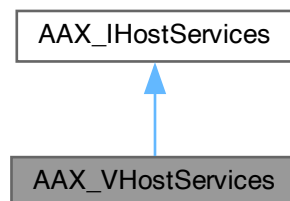
14.155 AAX_VHostServices Class Reference

```
#include <AAX_VHostServices.h>
```

Inheritance diagram for AAX_VHostServices:



Collaboration diagram for AAX_VHostServices:



14.155.1 Description

Version-managed concrete [AAX_IHostServices](#) class.

Public Member Functions

- [AAX_VHostServices](#) ([IACFUnknown](#) *pUnkHost)
- [~AAX_VHostServices](#) ()
- [AAX_Result HandleAssertFailure](#) (const char *iFile, int32_t iLine, const char *iNote, int32_t iFlags) const [AAX_OVERRIDE](#)
Handle an assertion failure.
- [AAX_Result Trace](#) (int32_t iPriority, const char *iMessage) const [AAX_OVERRIDE](#)
Log a trace message.
- [AAX_Result StackTrace](#) (int32_t iTracePriority, int32_t iStackTracePriority, const char *iMessage) const [AAX_OVERRIDE](#)
Log a trace message or a stack trace.

Public Member Functions inherited from [AAX_IHostServices](#)

- virtual [~AAX_IHostServices](#) ()
- virtual [AAX_Result HandleAssertFailure](#) (const char *iFile, int32_t iLine, const char *iNote, int32_t iFlags) const =0
Handle an assertion failure.
- virtual [AAX_Result Trace](#) (int32_t iPriority, const char *iMessage) const =0
Log a trace message.
- virtual [AAX_Result StackTrace](#) (int32_t iTracePriority, int32_t iStackTracePriority, const char *iMessage) const =0
Log a trace message or a stack trace.

14.155.2 Constructor & Destructor Documentation

14.155.2.1 [AAX_VHostServices\(\)](#)

```
AAX_VHostServices::AAX_VHostServices (
    IACFUnknown * pUnkHost )
```

14.155.2.2 [~AAX_VHostServices\(\)](#)

```
AAX_VHostServices::~~AAX_VHostServices ( )
```

14.155.3 Member Function Documentation

14.155.3.1 HandleAssertFailure()

```
AAX_Result AAX_VHostServices::HandleAssertFailure (
    const char * iFile,
    int32_t iLine,
    const char * iNote,
    int32_t iFlags ) const [virtual]
```

Handle an assertion failure.

Use this method to delegate assertion failure handling to the host

Use `iFlags` to request that specific behavior be included when handling the failure. This request may not be fulfilled by the host, and absence of a flag does not preclude the host from using that behavior when handling the failure.

Parameters

in	<i>iFile</i>	The name of the file containing the assert check. Usually <code>__FILE__</code>
in	<i>iLine</i>	The line number of the assert check. Usually <code>__LINE__</code>
in	<i>iNote</i>	Text to display related to the assert. Usually the condition which failed
in	<i>iFlags</i>	Bitfield of AAX_EAssertFlags to request specific handling behavior

Implements [AAX_IHostServices](#).

14.155.3.2 Trace()

```
AAX_Result AAX_VHostServices::Trace (
    int32_t iPriority,
    const char * iMessage ) const [virtual]
```

Log a trace message.

Parameters

in	<i>iPriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iMessage</i>	Message string to log

Implements [AAX_IHostServices](#).

14.155.3.3 StackTrace()

```
AAX_Result AAX_VHostServices::StackTrace (
    int32_t iTracePriority,
```

```
int32_t iStackTracePriority,
const char * iMessage ) const [virtual]
```

Log a trace message or a stack trace.

If the logging output filtering is set to include logs with `iStackTracePriority` then both the logging message and a stack trace will be emitted, regardless of `iTracePriority`.

If the logging output filtering is set to include logs with `iTracePriority` but to exclude logs with `iStackTracePriority` then this will emit a normal log with no stack trace.

Parameters

in	<i>iTracePriority</i>	Priority of the trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iStackTracePriority</i>	Priority of the stack trace, used for log filtering. One of kAAX_Trace_Priority_Low , kAAX_Trace_Priority_Normal , kAAX_Trace_Priority_High
in	<i>iMessage</i>	Message string to log

Implements [AAX_IHostServices](#).

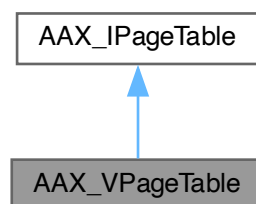
The documentation for this class was generated from the following file:

- [AAX_VHostServices.h](#)

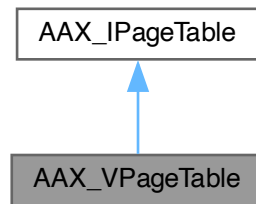
14.156 AAX_VPageTable Class Reference

```
#include <AAX_VPageTable.h>
```

Inheritance diagram for AAX_VPageTable:



Collaboration diagram for AAX_VPageTable:



14.156.1 Description

Version-managed concrete [AAX_IPageTable](#) class.

Public Member Functions

- [AAX_VPageTable](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VPageTable](#) () [AAX_OVERRIDE](#)
- [AAX_Result Clear](#) () [AAX_OVERRIDE](#)
Clears all parameter mappings from the table.
- [AAX_Result Empty](#) ([AAX_CBoolean](#) &oEmpty) const [AAX_OVERRIDE](#)
Indicates whether the table is empty.
- [AAX_Result GetNumPages](#) (int32_t &oNumPages) const [AAX_OVERRIDE](#)
Get the number of pages currently in this table.
- [AAX_Result InsertPage](#) (int32_t iPage) [AAX_OVERRIDE](#)
Insert a new empty page before the page at index iPage.
- [AAX_Result RemovePage](#) (int32_t iPage) [AAX_OVERRIDE](#)
Remove the page at index iPage.
- [AAX_Result GetNumMappedParameterIDs](#) (int32_t iPage, int32_t &oNumParameterIdentifiers) const [AAX_OVERRIDE](#)
Returns the total number of parameter IDs which are mapped to a page.
- [AAX_Result ClearMappedParameter](#) (int32_t iPage, int32_t iIndex) [AAX_OVERRIDE](#)
Clear the parameter at a particular index in this table.
- [AAX_Result GetMappedParameterID](#) (int32_t iPage, int32_t iIndex, [AAX_IString](#) &oParameterIdentifier) const [AAX_OVERRIDE](#)
Get the parameter identifier which is currently mapped to an index in this table.
- [AAX_Result MapParameterID](#) ([AAX_CParamID](#) iParameterIdentifier, int32_t iPage, int32_t iIndex) [AAX_OVERRIDE](#)
Map a parameter to this table.
- [AAX_Result GetNumParametersWithNameVariations](#) (int32_t &oNumParameterIdentifiers) const [AAX_OVERRIDE](#)
- [AAX_Result GetNameVariationParameterIDAtIndex](#) (int32_t iIndex, [AAX_IString](#) &oParameterIdentifier) const [AAX_OVERRIDE](#)
- [AAX_Result GetNumNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, int32_t &oNumVariations) const [AAX_OVERRIDE](#)

- [AAX_Result GetParameterNameVariationAtIndex](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, [int32_t](#) iIndex, [AAX_IString](#) &oNameVariation, [int32_t](#) &oLength) const [AAX_OVERRIDE](#)
- [AAX_Result GetParameterNameVariationOfLength](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, [int32_t](#) iLength, [AAX_IString](#) &oNameVariation) const [AAX_OVERRIDE](#)
- [AAX_Result ClearParameterNameVariations](#) () [AAX_OVERRIDE](#)
- [AAX_Result ClearNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier) [AAX_OVERRIDE](#)
- [AAX_Result SetParameterNameVariation](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, const [AAX_IString](#) &iNameVariation, [int32_t](#) iLength) [AAX_OVERRIDE](#)
- const [IACFUnknown](#) * [AsUnknown](#) () const
- [IACFUnknown](#) * [AsUnknown](#) ()
- bool [IsSupported](#) () const

Public Member Functions inherited from [AAX_IPageTable](#)

- virtual [~AAX_IPageTable](#) ()
Virtual destructor.
- virtual [AAX_Result Clear](#) ()=0
Clears all parameter mappings from the table.
- virtual [AAX_Result Empty](#) ([AAX_CBoolean](#) &oEmpty) const =0
Indicates whether the table is empty.
- virtual [AAX_Result GetNumPages](#) ([int32_t](#) &oNumPages) const =0
Get the number of pages currently in this table.
- virtual [AAX_Result InsertPage](#) ([int32_t](#) iPage)=0
Insert a new empty page before the page at index iPage.
- virtual [AAX_Result RemovePage](#) ([int32_t](#) iPage)=0
Remove the page at index iPage.
- virtual [AAX_Result GetNumMappedParameterIDs](#) ([int32_t](#) iPage, [int32_t](#) &oNumParameterIdentifiers) const =0
Returns the total number of parameter IDs which are mapped to a page.
- virtual [AAX_Result ClearMappedParameter](#) ([int32_t](#) iPage, [int32_t](#) iIndex)=0
Clear the parameter at a particular index in this table.
- virtual [AAX_Result GetMappedParameterID](#) ([int32_t](#) iPage, [int32_t](#) iIndex, [AAX_IString](#) &oParameterIdentifier) const =0
Get the parameter identifier which is currently mapped to an index in this table.
- virtual [AAX_Result MapParameterID](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, [int32_t](#) iPage, [int32_t](#) iIndex)=0
Map a parameter to this table.
- virtual [AAX_Result GetNumParametersWithNameVariations](#) ([int32_t](#) &oNumParameterIdentifiers) const =0
- virtual [AAX_Result GetNameVariationParameterIDAtIndex](#) ([int32_t](#) iIndex, [AAX_IString](#) &oParameterIdentifier) const =0
- virtual [AAX_Result GetNumNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, [int32_t](#) &oNumVariations) const =0
- virtual [AAX_Result GetParameterNameVariationAtIndex](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, [int32_t](#) iIndex, [AAX_IString](#) &oNameVariation, [int32_t](#) &oLength) const =0
- virtual [AAX_Result GetParameterNameVariationOfLength](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, [int32_t](#) iLength, [AAX_IString](#) &oNameVariation) const =0
- virtual [AAX_Result ClearParameterNameVariations](#) ()=0
- virtual [AAX_Result ClearNameVariationsForParameter](#) ([AAX_CPageTableParamID](#) iParameterIdentifier)=0
- virtual [AAX_Result SetParameterNameVariation](#) ([AAX_CPageTableParamID](#) iParameterIdentifier, const [AAX_IString](#) &iNameVariation, [int32_t](#) iLength)=0

14.156.2 Constructor & Destructor Documentation

14.156.2.1 AAX_VPageTable()

```
AAX_VPageTable::AAX_VPageTable (
    IACFUnknown * pUnknown )
```

14.156.2.2 ~AAX_VPageTable()

```
AAX_VPageTable::~~AAX_VPageTable ( )
```

14.156.3 Member Function Documentation

14.156.3.1 Clear()

```
AAX_Result AAX_VPageTable::Clear ( ) [virtual]
```

Clears all parameter mappings from the table.

This method does not clear any parameter name variations from the table. For that, use [AAX_IPageTable::ClearParameterNameVariations](#) or [AAX_IPageTable::ClearNameVariationsForParameter\(\)](#)

Implements [AAX_IPageTable](#).

14.156.3.2 Empty()

```
AAX_Result AAX_VPageTable::Empty (
    AAX_CBoolean & oEmpty ) const [virtual]
```

Indicates whether the table is empty.

A table is empty if it contains no pages. A table which contains pages but no parameter assignments is not empty. A table which has associated parameter name variations but no pages is empty.

Parameters

out	<i>oEmpty</i>	true if this table is empty
-----	---------------	-----------------------------

Implements [AAX_IPageTable](#).

14.156.3.3 GetNumPages()

```
AAX_Result AAX_VPageTable::GetNumPages (
    int32_t & oNumPages ) const [virtual]
```

Get the number of pages currently in this table.

Parameters

out	<i>oNumPages</i>	The number of pages which are present in the page table. Some pages might not contain any parameter assignments.
-----	------------------	--

Implements [AAX_IPageTable](#).

14.156.3.4 InsertPage()

```
AAX_Result AAX_VPageTable::InsertPage (
    int32_t iPage ) [virtual]
```

Insert a new empty page before the page at index *iPage*.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the total number of pages

Parameters

in	<i>iPage</i>	The insertion point page index
----	--------------	--------------------------------

Implements [AAX_IPageTable](#).

14.156.3.5 RemovePage()

```
AAX_Result AAX_VPageTable::RemovePage (
    int32_t iPage ) [virtual]
```

Remove the page at index *iPage*.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

Parameters

in	<i>iPage</i>	The target page index
----	--------------	-----------------------

Implements [AAX_IPageTable](#).

14.156.3.6 GetNumMappedParameterIDs()

```
AAX_Result AAX_VPageTable::GetNumMappedParameterIDs (
    int32_t iPage,
    int32_t & oNumParameterIdentifiers ) const [virtual]
```

Returns the total number of parameter IDs which are mapped to a page.

Note

The number of mapped parameter IDs does not correspond to the actual slot indices of the parameter assignments. For example, a page could have three total parameter assignments with parameters mapped to slots 2, 4, and 6.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

Parameters

in	<i>iPage</i>	The target page index
out	<i>oNumParameterIdentifiers</i>	The number of parameter identifiers which are mapped to the target page

Implements [AAX_IPageTable](#).

14.156.3.7 ClearMappedParameter()

```
AAX_Result AAX_VPageTable::ClearMappedParameter (
    int32_t iPage,
    int32_t iIndex ) [virtual]
```

Clear the parameter at a particular index in this table.

Returns

[AAX_SUCCESS](#) even if no parameter was mapped at the given index (the index is still clear)

Parameters

in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page

Implements [AAX_IPageTable](#).

14.156.3.8 GetMappedParameterID()

```
AAX_Result AAX_VPageTable::GetMappedParameterID (
    int32_t iPage,
    int32_t iIndex,
    AAX_IString & oParameterIdentifier ) const [virtual]
```

Get the parameter identifier which is currently mapped to an index in this table.

Returns

[AAX_ERROR_INVALID_ARGUMENT](#) if no parameter is mapped at the specified page/index location

Parameters

in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page
out	<i>oParameterIdentifier</i>	The identifier used for the mapped parameter in the page table (may be parameter name or ID)

Implements [AAX_IPageTable](#).

14.156.3.9 MapParameterID()

```
AAX_Result AAX_VPageTable::MapParameterID (
    AAX_CParamID iParameterIdentifier,
    int32_t iPage,
    int32_t iIndex ) [virtual]
```

Map a parameter to this table.

If *iParameterIdentifier* is an empty string then the parameter assignment will be cleared

Returns

[AAX_ERROR_NULL_ARGUMENT](#) if *iParameterIdentifier* is null

[AAX_ERROR_INVALID_ARGUMENT](#) if *iPage* is greater than the index of the last existing page

[AAX_ERROR_INVALID_ARGUMENT](#) if *iIndex* is negative

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter which will be mapped
in	<i>iPage</i>	The target page index
in	<i>iIndex</i>	The target parameter slot index within the target page

Implements [AAX_IPageTable](#).

14.156.3.10 GetNumParametersWithNameVariations()

```
AAX_Result AAX_VPageTable::GetNumParametersWithNameVariations (
    int32_t & oNumParameterIdentifiers ) const [virtual]
```

Get the number of parameters with name variations defined for the current table type

Provides the number of parameters with `lt;ControlNameVariationslt;` which are explicitly defined for the current page table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNameVariationParameterIDAtIndex\(\)](#)

Parameters

out	<i>oNumParameterIdentifiers</i>	The number of parameters with name variations explicitly associated with the current table type.
-----	---------------------------------	--

Implements [AAX_IPageTable](#).

14.156.3.11 GetNameVariationParameterIDAtIndex()

```
AAX_Result AAX_VPageTable::GetNameVariationParameterIDAtIndex (
    int32_t iIndex,
    AAX_IString & oParameterIdentifier ) const [virtual]
```

Get the identifier for a parameter with name variations defined for the current table type

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNumParametersWithNameVariations\(\)](#)

Parameters

in	<i>iIndex</i>	The target parameter index within the list of parameters with explicit name variations defined for this table type.
out	<i>oParameterIdentifier</i>	The identifier used for the parameter in the page table name variations list (may be parameter name or ID)

Implements [AAX_IPageTable](#).

14.156.3.12 GetNumNameVariationsForParameter()

```
AAX_Result AAX_VPageTable::GetNumNameVariationsForParameter (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t & oNumVariations ) const [virtual]
```

Get the number of name variations defined for a parameter

Provides the number of `lt;ControlNameVariationslt;` which are explicitly defined for `iParameterIdentifier` for the current page table type. No fallback logic is used to resolve this to the list of variations which would actually be used for an attached control surface if no explicit variations are defined for the current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetParameterNameVariationAtIndex\(\)](#)

Returns

[AAX_SUCCESS](#) and provides zero to `oNumVariations` if `iParameterIdentifier` is not found

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
out	<i>oNumVariations</i>	The number of name variations which are defined for this parameter and explicitly associated with the current table type.

Implements [AAX_IPageTable](#).

14.156.3.13 GetParameterNameVariationAtIndex()

```
AAX_Result AAX_VPageTable::GetParameterNameVariationAtIndex (
    AAX_CPageTableParamID iParameterIdentifier,
```

```
int32_t iIndex,
AAX_IString & oNameVariation,
int32_t & oLength ) const [virtual]
```

Get a parameter name variation from the page table

Only returns `lt;ControlNameVariationslt;` which are explicitly defined for the current page table type. No fallback logic is used to resolve this to the abbreviation which would actually be shown on an attached control surface if no explicit variation is defined for the current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetNumNameVariationsForParameter\(\)](#)

See also

- [AAX_IPageTable::GetParameterNameVariationOfLength\(\)](#)

Returns

[AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME](#) if no suitable variation is defined for this table
[AAX_ERROR_ARGUMENT_OUT_OF_RANGE](#) if `iIndex` is out of range

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iIndex</i>	Index of the name variation
out	<i>oNameVariation</i>	The name variation, if one is explicitly defined for this table type
out	<i>oLength</i>	The length value for this name variation. This corresponds to the variation's <code>sz</code> attribute in the page table XML and may be different from the string length of <code>iNameVariation</code> .

Implements [AAX_IPageTable](#).

14.156.3.14 GetParameterNameVariationOfLength()

```
AAX_Result AAX_VPageTable::GetParameterNameVariationOfLength (
    AAX_CPageTableParamID iParameterIdentifier,
    int32_t iLength,
    AAX_IString & oNameVariation ) const [virtual]
```

Get a parameter name variation of a particular length from the page table

Only returns `lt;ControlNameVariationslt;` which are explicitly defined of `iLength` for the current page table type. No fallback logic is used to resolve this to the abbreviation which would actually be shown on an attached control surface if no explicit variation is defined for the specified length or current table type.

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

- [AAX_IPageTable::GetParameterNameVariationAtIndex\(\)](#)

Returns

[AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME](#) if no suitable variation is defined for this table

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iLength</i>	The variation length to check, i.e. the <i>sz</i> attribute for the name variation in the page table XML
out	<i>oNameVariation</i>	The name variation, if one is explicitly defined for this table type and <i>iLength</i>

Implements [AAX_IPageTable](#).

14.156.3.15 ClearParameterNameVariations()

```
AAX\_Result AAX_VPageTable::ClearParameterNameVariations ( ) [virtual]
```

Clears all name variations for the current page table type

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

[AAX_IPageTable::Clear\(\)](#)

[AAX_IPageTable::ClearNameVariationsForParameter\(\)](#)

Implements [AAX_IPageTable](#).

14.156.3.16 ClearNameVariationsForParameter()

```
AAX_Result AAX_VPageTable::ClearNameVariationsForParameter (
    AAX_CPageTableParamID iParameterIdentifier ) [virtual]
```

Clears all name variations for a single parameter for the current page table type

Warning

This will invalidate the list of parameter name variations indices, i.e. the parameter identifier associated with each index by [AAX_IPageTable::GetNameVariationParameterIDAtIndex\(\)](#)

Note

Normally parameter name variations are only used with the 'PgTL' table type

See also

[AAX_IPageTable::Clear\(\)](#)

[AAX_IPageTable::ClearParameterNameVariations\(\)](#)

Returns

[AAX_SUCCESS](#) and provides zero to oNumVariations if iParameterIdentifier is not found

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
----	-----------------------------	----------------------------------

Implements [AAX_IPageTable](#).

14.156.3.17 SetParameterNameVariation()

```
AAX_Result AAX_VPageTable::SetParameterNameVariation (
    AAX_CPageTableParamID iParameterIdentifier,
    const AAX_IString & iNameVariation,
    int32_t iLength ) [virtual]
```

Sets a name variation explicitly for the current page table type

This will add a new name variation or overwrite the existing name variation with the same length which is defined for the current table type.

Warning

If no name variation previously existed for this parameter then this will invalidate the list of parameter name variations indices, i.e. the parameter identifier associated with each index by [AAX_IPageTable::GetNameVariationParameterIDAtIndex\(\)](#)

Note

Normally parameter name variations are only used with the 'PgTL' table type

Returns

AAX_ERROR_INVALID_ARGUMENT if `iNameVariation` is empty or if `iLength` is less than zero

Parameters

in	<i>iParameterIdentifier</i>	The identifier for the parameter
in	<i>iNameVariation</i>	The new parameter name variation
in	<i>iLength</i>	The length value for this name variation. This corresponds to the variation's <code>sz</code> attribute in the page table XML and is not required to match the length of <code>iNameVariation</code> .

Implements [AAX_IPageTable](#).

14.156.3.18 AsUnknown() [1/2]

```
const IACFUnknown * AAX_VPageTable::AsUnknown ( ) const [inline]
```

Returns the latest supported versioned ACF interface (e.g. an [AAX_IACFPageTable](#)) which is wrapped by this [AAX_IPageTable](#)

14.156.3.19 AsUnknown() [2/2]

```
IACFUnknown * AAX_VPageTable::AsUnknown ( ) [inline]
```

Returns the latest supported versioned ACF interface (e.g. an [AAX_IACFPageTable](#)) which is wrapped by this [AAX_IPageTable](#)

14.156.3.20 IsSupported()

```
bool AAX_VPageTable::IsSupported ( ) const [inline]
```

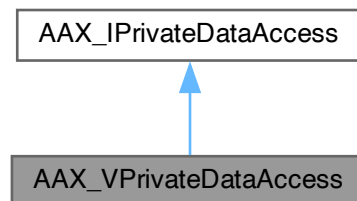
The documentation for this class was generated from the following file:

- [AAX_VPageTable.h](#)

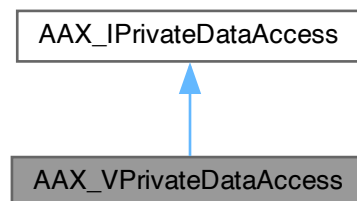
14.157 AAX_VPrivateDataAccess Class Reference

```
#include <AAX_VPrivateDataAccess.h>
```

Inheritance diagram for AAX_VPrivateDataAccess:



Collaboration diagram for AAX_VPrivateDataAccess:



14.157.1 Description

Version-managed concrete [AAX_IPrivateDataAccess](#) class.

Public Member Functions

- [AAX_VPrivateDataAccess](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VPrivateDataAccess](#) () [AAX_OVERRIDE](#)
- [AAX_Result ReadPortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, void *outBuffer) [AAX_OVERRIDE](#)
Read data directly from DSP at the given port.
- [AAX_Result WritePortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, const void *inBuffer) [AAX_OVERRIDE](#)
Write data directly to DSP at the given port.

Public Member Functions inherited from [AAX_IPrivateDataAccess](#)

- virtual [~AAX_IPrivateDataAccess](#) ()
- virtual [AAX_Result ReadPortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, void *outBuffer)=0
Read data directly from DSP at the given port.
- virtual [AAX_Result WritePortDirect](#) ([AAX_CFieldIndex](#) inFieldIndex, const uint32_t inOffset, const uint32_t inSize, const void *inBuffer)=0
Write data directly to DSP at the given port.

14.157.2 Constructor & Destructor Documentation

14.157.2.1 AAX_VPrivateDataAccess()

```
AAX_VPrivateDataAccess::AAX_VPrivateDataAccess (
    IACFUnknown * pUnknown )
```

14.157.2.2 ~AAX_VPrivateDataAccess()

```
AAX_VPrivateDataAccess::~~AAX_VPrivateDataAccess ( )
```

14.157.3 Member Function Documentation

14.157.3.1 ReadPortDirect()

```
AAX\_Result AAX_VPrivateDataAccess::ReadPortDirect (
    AAX\_CFieldIndex inFieldIndex,
    const uint32_t inOffset,
    const uint32_t inSize,
    void * outBuffer ) [virtual]
```

Read data directly from DSP at the given port.

Note

Blocking

Parameters

in	<i>inFieldIndex</i>	The port to read from.
in	<i>inOffset</i>	Offset into data to start reading.
in	<i>inSize</i>	Amount of data to read (in bytes).
out	<i>outBuffer</i>	Pointer to storage for data to be read into.

Implements [AAX_IPrivateDataAccess](#).

14.157.3.2 WritePortDirect()

```
AAX_Result AAX_VPrivateDataAccess::WritePortDirect (
    AAX_CFieldIndex inFieldIndex,
    const uint32_t inOffset,
    const uint32_t inSize,
    const void * inBuffer ) [virtual]
```

Write data directly to DSP at the given port.

Note

Blocking

Parameters

in	<i>inFieldIndex</i>	The port to write to.
in	<i>inOffset</i>	Offset into data to begin writing.
in	<i>inSize</i>	Amount of data to write (in bytes).
in	<i>inBuffer</i>	Pointer to data being written.

Implements [AAX_IPrivateDataAccess](#).

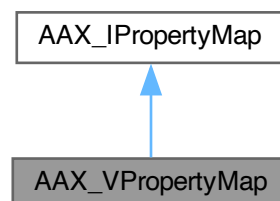
The documentation for this class was generated from the following file:

- [AAX_VPrivateDataAccess.h](#)

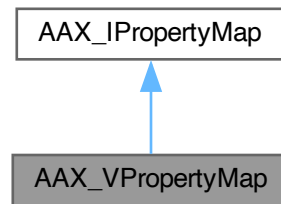
14.158 AAX_VPropertyMap Class Reference

```
#include <AAX_VPropertyMap.h>
```

Inheritance diagram for AAX_VPropertyMap:



Collaboration diagram for AAX_VPropertyMap:



14.158.1 Description

Version-managed concrete [AAX_IPropertyMap](#) class.

Public Member Functions

- `~AAX_VPropertyMap` (void) [AAX_OVERRIDE](#)
- `AAX_CBoolean GetProperty` ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) *outValue) const [AAX_OVERRIDE](#)
Get a property value from a property map.
- `AAX_CBoolean GetPointerProperty` ([AAX_EProperty](#) inProperty, const void **outValue) const [AAX_OVERRIDE](#)
Get a property value from a property map with a pointer-sized value.
- `AAX_Result AddProperty` ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) inValue) [AAX_OVERRIDE](#)
Add a property to a property map.
- `AAX_Result AddPointerProperty` ([AAX_EProperty](#) inProperty, const void *inValue) [AAX_OVERRIDE](#)
Add a property to a property map with a pointer-sized value.
- `AAX_Result AddPointerProperty` ([AAX_EProperty](#) inProperty, const char *inValue) [AAX_OVERRIDE](#)
Add a property to a property map with a pointer-sized value.
- `AAX_Result RemoveProperty` ([AAX_EProperty](#) inProperty) [AAX_OVERRIDE](#)
Remove a property from a property map.
- `AAX_Result AddPropertyWithIDArray` ([AAX_EProperty](#) inProperty, const [AAX_SPluginIdentifierTriad](#) *inPluginIDs, uint32_t inNumPluginIDs) [AAX_OVERRIDE](#)
Add an array of plug-in IDs to a property map.
- `AAX_CBoolean GetPropertyWithIDArray` ([AAX_EProperty](#) inProperty, const [AAX_SPluginIdentifierTriad](#) **outPluginIDs, uint32_t *outNumPluginIDs) const [AAX_OVERRIDE](#)
Get an array of plug-in IDs from a property map.
- `IACFUnknown * GetUnknown` () [AAX_OVERRIDE](#)

Public Member Functions inherited from AAX_IPropertyMap

- virtual [~AAX_IPropertyMap](#) ()
- virtual [AAX_CBoolean](#) [GetProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) *outValue) const =0
Get a property value from a property map.
- virtual [AAX_CBoolean](#) [GetPointerProperty](#) ([AAX_EProperty](#) inProperty, const void **outValue) const =0
Get a property value from a property map with a pointer-sized value.
- virtual [AAX_Result](#) [AddProperty](#) ([AAX_EProperty](#) inProperty, [AAX_CPropertyValue](#) inValue)=0
Add a property to a property map.
- virtual [AAX_Result](#) [AddPointerProperty](#) ([AAX_EProperty](#) inProperty, const void *inValue)=0
Add a property to a property map with a pointer-sized value.
- virtual [AAX_Result](#) [AddPointerProperty](#) ([AAX_EProperty](#) inProperty, const char *inValue)=0
Add a property to a property map with a pointer-sized value.
- virtual [AAX_Result](#) [RemoveProperty](#) ([AAX_EProperty](#) inProperty)=0
Remove a property from a property map.
- virtual [AAX_Result](#) [AddPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPlugInIdentifierTriad](#) *inPluginIDs, uint32_t inNumPluginIDs)=0
Add an array of plug-in IDs to a property map.
- virtual [AAX_CBoolean](#) [GetPropertyWithIDArray](#) ([AAX_EProperty](#) inProperty, const [AAX_SPlugInIdentifierTriad](#) **outPluginIDs, uint32_t *outNumPluginIDs) const =0
Get an array of plug-in IDs from a property map.
- virtual [IACFUnknown](#) * [GetIUnknown](#) ()=0

Static Public Member Functions

- static [AAX_VPropertyMap](#) * [Create](#) ([IACFUnknown](#) *inComponentFactory)
inComponentFactory must support IID_IACFComponentFactory - otherwise NULL is returned
- static [AAX_VPropertyMap](#) * [Acquire](#) ([IACFUnknown](#) *inPropertyMapUnknown)
inPropertyMapUnknown must support at least one AAX_IPropertyMap interface - otherwise an AAX_VPropertyMap object with no backing interface is returned

14.158.2 Constructor & Destructor Documentation**14.158.2.1 ~AAX_VPropertyMap()**

```
AAX_VPropertyMap::~~AAX_VPropertyMap (
    void )
```

14.158.3 Member Function Documentation

14.158.3.1 Create()

```
static AAX_VPropertyMap * AAX_VPropertyMap::Create (
    IACFUnknown * inComponentFactory ) [static]
```

`inComponentFactory` must support `IID_IACFComponentFactory` - otherwise NULL is returned

14.158.3.2 Acquire()

```
static AAX_VPropertyMap * AAX_VPropertyMap::Acquire (
    IACFUnknown * inPropertyMapUnknown ) [static]
```

`inPropertyMapUnknown` must support at least one `AAX_IPropertyMap` interface - otherwise an `AAX_VPropertyMap` object with no backing interface is returned

14.158.3.3 GetProperty()

```
AAX_CBoolean AAX_VPropertyMap::GetProperty (
    AAX_EProperty inProperty,
    AAX_CPropertyValue * outValue ) const [virtual]
```

Get a property value from a property map.

Returns true if the selected property is supported, false if it is not

Parameters

in	<i>inProperty</i>	The property ID
out	<i>outValue</i>	The property value

Implements `AAX_IPropertyMap`.

14.158.3.4 GetPointerProperty()

```
AAX_CBoolean AAX_VPropertyMap::GetPointerProperty (
    AAX_EProperty inProperty,
    const void ** outValue ) const [virtual]
```

Get a property value from a property map with a pointer-sized value.

Returns true if the selected property is supported, false if it is not

Parameters

in	<i>inProperty</i>	The property ID
out	<i>outValue</i>	The property value

Implements [AAX_IPropertyMap](#).

14.158.3.5 AddProperty()

```
AAX_Result AAX_VPropertyMap::AddProperty (
    AAX_EProperty inProperty,
    AAX_CPropertyValue inValue ) [virtual]
```

Add a property to a property map.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

Implements [AAX_IPropertyMap](#).

14.158.3.6 AddPointerProperty() [1/2]

```
AAX_Result AAX_VPropertyMap::AddPointerProperty (
    AAX_EProperty inProperty,
    const void * inValue ) [virtual]
```

Add a property to a property map with a pointer-sized value.

Use this method to add properties which require a pointer-sized value. Do not use this method to add a property unless a pointer-sized value is explicitly specified in the property documentation.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

Implements [AAX_IPropertyMap](#).

14.158.3.7 AddPointerProperty() [2/2]

```
AAX_Result AAX_VPropertyMap::AddPointerProperty (
    AAX_EProperty inProperty,
    const char * inValue ) [virtual]
```

Add a property to a property map with a pointer-sized value.

Use this method to add properties which require a pointer-sized value. Do not use this method to add a property unless a pointer-sized value is explicitly specified in the property documentation.

Note

This method may return an error if adding the property was unsuccessful. If there is a failure when adding a required property then registration of the relevant description element must be abandoned and the plug-in's description logic should proceed to the next element.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inValue</i>	

Implements [AAX_IPropertyMap](#).

14.158.3.8 RemoveProperty()

```
AAX_Result AAX_VPropertyMap::RemoveProperty (
    AAX_EProperty inProperty ) [virtual]
```

Remove a property from a property map.

Parameters

in	<i>inProperty</i>	The property ID.
----	-------------------	------------------

Implements [AAX_IPropertyMap](#).

14.158.3.9 AddPropertyWithIDArray()

```
AAX_Result AAX_VPropertyMap::AddPropertyWithIDArray (
    AAX_EProperty inProperty,
    const AAX_SPlugInIdentifierTriad * inPluginIDs,
    uint32_t inNumPluginIDs ) [virtual]
```

Add an array of plug-in IDs to a property map.

Parameters

in	<i>inProperty</i>	The property ID.
in	<i>inPluginIDs</i>	An array of AAX_SPlugInIdentifierTriad
in	<i>inNumPluginIDs</i>	The length of iPluginIDs

Implements [AAX_IPropertyMap](#).

14.158.3.10 GetPropertyWithIDArray()

```
AAX_CBoolean AAX_VPropertyMap::GetPropertyWithIDArray (
    AAX_EProperty inProperty,
    const AAX_SPlugInIdentifierTriad ** outPluginIDs,
    uint32_t * outNumPluginIDs ) const [virtual]
```

Get an array of plug-in IDs from a property map.

Parameters

in	<i>inProperty</i>	The property ID.
out	<i>outPluginIDs</i>	A pointer that will be set to reference an array of AAX_SPlugInIdentifierTriad
in	<i>outNumPluginIDs</i>	The length of oPluginIDs

Implements [AAX_IPropertyMap](#).

14.158.3.11 GetIUnknown()

```
IACFUnknown * AAX_VPropertyMap::GetIUnknown ( ) [virtual]
```

Returns the most up-to-date underlying interface

Implements [AAX_IPropertyMap](#).

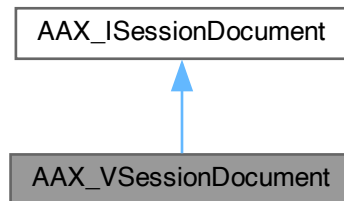
The documentation for this class was generated from the following file:

- [AAX_VPropertyMap.h](#)

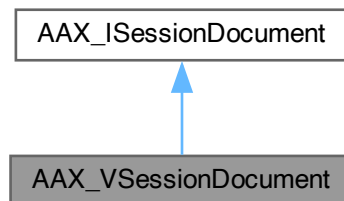
14.159 AAX_VSessionDocument Class Reference

```
#include <AAX_VSessionDocument.h>
```

Inheritance diagram for AAX_VSessionDocument:



Collaboration diagram for AAX_VSessionDocument:



Classes

- class [VTempoMap](#)

Public Member Functions

- [AAX_VSessionDocument](#) ([IACFUnknown](#) *iUnknown)
- [~AAX_VSessionDocument](#) () [AAX_OVERRIDE](#)
- void [Clear](#) ()
Release all interface references.
- bool [Valid](#) () const [AAX_OVERRIDE](#)
Check whether this session document is valid.
- [std::unique_ptr< AAX_ISessionDocument::TempoMap const >](#) [GetTempoMap](#) () [AAX_OVERRIDE](#)
Get a copy of the document's tempo map.
- [AAX_Result](#) [GetDocumentData](#) ([AAX_DocumentData_UID](#) const &inDataType, [IACFUnknown](#) **outData) [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_ISessionDocument](#)

- virtual [~AAX_ISessionDocument](#) ()=default
- virtual bool [Valid](#) () const =0
Check whether this session document is valid.
- virtual std::unique_ptr< [TempoMap](#) const > [GetTempoMap](#) ()=0
Get a copy of the document's tempo map.
- virtual [AAX_Result](#) [GetDocumentData](#) ([AAX_DocumentData_UID](#) const &inDataType, [IACFUnknown](#) **outData)=0

14.159.1 Constructor & Destructor Documentation**14.159.1.1 [AAX_VSessionDocument](#)()**

```
AAX_VSessionDocument::AAX_VSessionDocument (
    IACFUnknown * iUnknown ) [explicit]
```

14.159.1.2 [~AAX_VSessionDocument](#)()

```
AAX_VSessionDocument::~~AAX_VSessionDocument ( )
```

14.159.2 Member Function Documentation**14.159.2.1 [Clear](#)()**

```
void AAX_VSessionDocument::Clear ( )
```

Release all interface references.

14.159.2.2 [Valid](#)()

```
bool AAX_VSessionDocument::Valid ( ) const [virtual]
```

Check whether this session document is valid.

Implements [AAX_ISessionDocument](#).

14.159.2.3 GetTempoMap()

```
std::unique_ptr< AAX_I_SessionDocument::TempoMap const > AAX_VSessionDocument::GetTempoMap ( )
[virtual]
```

Get a copy of the document's tempo map.

Returns

A TempoMap interface representing a copy of the current tempo map.

`nullptr` if the host does not support tempo map data or if an error occurred.

Implements [AAX_I_SessionDocument](#).

14.159.2.4 GetDocumentData()

```
AAX_Result AAX_VSessionDocument::GetDocumentData (
    AAX_DocumentData_UID const & inDataType,
    IACFUnknown ** outData ) [virtual]
```

Get document data of a generic type

Similar to `QueryInterface()` but uses a data type identifier rather than a true IID

The provided interface has already had a reference added, so be careful not to add an additional reference:

```
ACFPtr<MyType> ptr;
IACFUnknown * docDataPtr(nullptr);
if (AAX_SUCCESS == doc->GetDocumentData(dataUID, &docDataPtr) && docDataPtr) {
    ptr.attach(std::static_cast<MyType*>(docDataPtr)); // attach does not AddRef
}
```

Parameters

in	<i>inDataType</i>	The type of the document data requested
out	<i>outData</i>	An interface providing the requested data, or <code>nullptr</code> if the host does not support or cannot provide the requested data type. The reference count has been incremented on this object on behalf of the caller, so the caller must not add an additional reference count and must decrement the reference count on this object to release it. For information about which interface to expect for each requested data type, see the documentation for that data type.

Implements [AAX_I_SessionDocument](#).

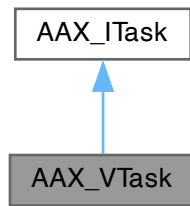
The documentation for this class was generated from the following file:

- [AAX_VSessionDocument.h](#)

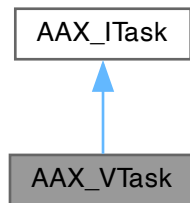
14.160 AAX_VTask Class Reference

```
#include <AAX_VTask.h>
```

Inheritance diagram for AAX_VTask:



Collaboration diagram for AAX_VTask:



14.160.1 Description

Version-managed concrete [AAX_ITask](#).

Public Member Functions

- [AAX_VTask](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VTask](#) () [AAX_OVERRIDE](#)
- [AAX_Result](#) [GetType](#) ([AAX_CTypeID](#) *oType) const [AAX_OVERRIDE](#)
- [AAX_IACFDataBuffer](#) const * [GetArgumentOfType](#) ([AAX_CTypeID](#) iType) const [AAX_OVERRIDE](#)
- [AAX_Result](#) [SetProgress](#) (float iProgress) [AAX_OVERRIDE](#)
- float [GetProgress](#) () const [AAX_OVERRIDE](#)
- [AAX_Result](#) [AddResult](#) ([AAX_IACFDataBuffer](#) const *iResult) [AAX_OVERRIDE](#)
Attach result data to this task.
- [AAX_ITask](#) * [SetDone](#) ([AAX_TaskCompletionStatus](#) iStatus) [AAX_OVERRIDE](#)
Inform the host that the task is completed.

Public Member Functions inherited from [AAX_ITask](#)

- virtual [~AAX_ITask](#) ()=default
- virtual [AAX_Result](#) [GetType](#) ([AAX_CTypeID](#) *oType) const =0
- virtual [AAX_IACFDataBuffer](#) const * [GetArgumentOfType](#) ([AAX_CTypeID](#) iType) const =0
- virtual [AAX_Result](#) [SetProgress](#) (float iProgress)=0
- virtual float [GetProgress](#) () const =0
- virtual [AAX_Result](#) [AddResult](#) ([AAX_IACFDataBuffer](#) const *iResult)=0
Attach result data to this task.
- virtual [AAX_ITask](#) * [SetDone](#) ([AAX_TaskCompletionStatus](#) iStatus)=0
Inform the host that the task is completed.

14.160.2 Constructor & Destructor Documentation

14.160.2.1 [AAX_VTask](#)()

```
AAX_VTask::AAX_VTask (
    IACFUnknown * pUnknown ) [explicit]
```

14.160.2.2 [~AAX_VTask](#)()

```
AAX_VTask::~~AAX_VTask ( )
```

14.160.3 Member Function Documentation

14.160.3.1 [GetType](#)()

```
AAX\_Result AAX\_VTask::GetType (
    AAX\_CTypeID * oType ) const [virtual]
```

An identifier defining the type of the requested task

Parameters

out	<i>oType</i>	The type of this task request
-----	--------------	-------------------------------

Implements [AAX_ITask](#).

14.160.3.2 GetArgumentOfType()

```
AAX_IACFDataBuffer const * AAX_VTask::GetArgumentOfType (
    AAX_CTypeID iType ) const [virtual]
```

Additional information defining the request, depending on the task type

Parameters

in	<i>iType</i>	The type of argument requested. Possible argument types, if any, and the resulting data buffer format must be defined per task type.
----	--------------	--

Returns

The requested argument data, or nullptr. This data buffer's type ID is expected to match *iType* . The caller takes ownership of this object.

Implements [AAX_ITask](#).

14.160.3.3 SetProgress()

```
AAX_Result AAX_VTask::SetProgress (
    float iProgress ) [virtual]
```

Inform the host about the current status of the task

Parameters

in	<i>iProgress</i>	A value between 0 (no progress) and 1 (complete)
----	------------------	--

Implements [AAX_ITask](#).

14.160.3.4 GetProgress()

```
float AAX_VTask::GetProgress ( ) const [virtual]
```

Returns the current progress

Implements [AAX_ITask](#).

14.160.3.5 AddResult()

```
AAX_Result AAX_VTask::AddResult (
    AAX_IACFDataBuffer const * iResult ) [virtual]
```

Attach result data to this task.

This can be called multiple times to add multiple types of results to a single task.

The host may process the result data immediately or may wait for the task to complete.

The plug-in is expected to release the data buffer upon making this call. At a minimum, the data buffer must not be changed after this call is made. See `ACFPtr::inArg()`

Parameters

in	<i>iResult</i>	A buffer containing the result data. Expected result types, if any, and their data buffer format must be defined per task type.
----	----------------	---

Implements [AAX_ITask](#).

14.160.3.6 SetDone()

```
AAX_ITask * AAX_VTask::SetDone (
    AAX_TaskCompletionStatus iStatus ) [virtual]
```

Inform the host that the task is completed.

If successful, returns a null pointer. Otherwise, returns a pointer back to the same object. This is the expected usage pattern:

```
// release the task on success, retain it on failure
myTask = myTask->SetDone(status);
```

Parameters

in	<i>iStatus</i>	The final status of the task. This indicates to the host whether or not the task was performed as requested.
----	----------------	--

Implements [AAX_ITask](#).

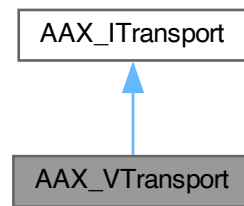
The documentation for this class was generated from the following file:

- [AAX_VTask.h](#)

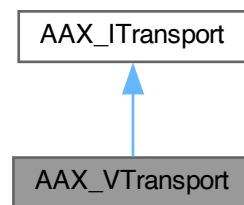
14.161 AAX_VTransport Class Reference

```
#include <AAX_VTransport.h>
```

Inheritance diagram for AAX_VTransport:



Collaboration diagram for AAX_VTransport:



14.161.1 Description

Version-managed concrete [AAX_ITransport](#) class.

Public Member Functions

- [AAX_VTransport](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VTransport](#) () [AAX_OVERRIDE](#)
- [AAX_Result GetCurrentTempo](#) (double *TempoBPM) const [AAX_OVERRIDE](#)
CALL: Gets the current tempo.
- [AAX_Result GetCurrentMeter](#) (int32_t *MeterNumerator, int32_t *MeterDenominator) const [AAX_OVERRIDE](#)
CALL: Gets the current meter.
- [AAX_Result IsTransportPlaying](#) (bool *isPlaying) const [AAX_OVERRIDE](#)
CALL: Indicates whether or not the transport is playing back.
- [AAX_Result GetCurrentTickPosition](#) (int64_t *TickPosition) const [AAX_OVERRIDE](#)
CALL: Gets the current tick position.
- [AAX_Result GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const [AAX_OVERRIDE](#)
CALL: Gets current information on loop playback.

- [AAX_Result GetCurrentNativeSampleLocation](#) (int64_t *SampleLocation) const [AAX_OVERRIDE](#)
CALL: Gets the current playback location of the native audio engine.
- [AAX_Result GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const [AAX_OVERRIDE](#)
CALL: Given an absolute sample position, gets the corresponding tick position.
- [AAX_Result GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t SampleLocation) const [AAX_OVERRIDE](#)
CALL: Given an absolute sample position, gets the corresponding bar and beat position.
- [AAX_Result GetTicksPerQuarter](#) (uint32_t *ticks) const [AAX_OVERRIDE](#)
CALL: Retrieves the number of ticks per quarter note.
- [AAX_Result GetCurrentTicksPerBeat](#) (uint32_t *ticks) const [AAX_OVERRIDE](#)
CALL: Retrieves the number of ticks per beat.
- [AAX_Result GetTimelineSelectionStartPosition](#) (int64_t *oSampleLocation) const [AAX_OVERRIDE](#)
CALL: Retrieves the absolute sample position of the beginning of the current transport selection.
- [AAX_Result GetTimeCodeInfo](#) ([AAX_EFrameRate](#) *oFrameRate, int32_t *oOffset) const [AAX_OVERRIDE](#)
CALL: Retrieves the current time code frame rate and offset.
- [AAX_Result GetFeetFramesInfo](#) ([AAX_EFeetFramesRate](#) *oFeetFramesRate, int64_t *oOffset) const [AAX_OVERRIDE](#)
CALL: Retrieves the current timecode feet/frames rate and offset.
- [AAX_Result IsMetronomeEnabled](#) (int32_t *isEnabled) const [AAX_OVERRIDE](#)
Sets isEnabled to true if the metronome is enabled.
- [AAX_Result GetHDTimeCodeInfo](#) ([AAX_EFrameRate](#) *oHDFrameRate, int64_t *oHDOffset) const [AAX_OVERRIDE](#)
CALL: Retrieves the current HD time code frame rate and offset.
- [AAX_Result GetTimelineSelectionEndPosition](#) (int64_t *oSampleLocation) const [AAX_OVERRIDE](#)
CALL: Retrieves the absolute sample position of the end of the current transport selection.
- [AAX_Result RequestTransportStart](#) () [AAX_OVERRIDE](#)
CALL: Request that the host transport start playback.
- [AAX_Result RequestTransportStop](#) () [AAX_OVERRIDE](#)
CALL: Request that the host transport stop playback.

Public Member Functions inherited from [AAX_ITransport](#)

- virtual [~AAX_ITransport](#) ()
Virtual destructor.
- virtual [AAX_Result GetCurrentTempo](#) (double *TempoBPM) const =0
CALL: Gets the current tempo.
- virtual [AAX_Result GetCurrentMeter](#) (int32_t *MeterNumerator, int32_t *MeterDenominator) const =0
CALL: Gets the current meter.
- virtual [AAX_Result IsTransportPlaying](#) (bool *isPlaying) const =0
CALL: Indicates whether or not the transport is playing back.
- virtual [AAX_Result GetCurrentTickPosition](#) (int64_t *TickPosition) const =0
CALL: Gets the current tick position.
- virtual [AAX_Result GetCurrentLoopPosition](#) (bool *bLooping, int64_t *LoopStartTick, int64_t *LoopEndTick) const =0
CALL: Gets current information on loop playback.
- virtual [AAX_Result GetCurrentNativeSampleLocation](#) (int64_t *SampleLocation) const =0
CALL: Gets the current playback location of the native audio engine.
- virtual [AAX_Result GetCustomTickPosition](#) (int64_t *oTickPosition, int64_t iSampleLocation) const =0
CALL: Given an absolute sample position, gets the corresponding tick position.
- virtual [AAX_Result GetBarBeatPosition](#) (int32_t *Bars, int32_t *Beats, int64_t *DisplayTicks, int64_t SampleLocation) const =0

- CALL: Given an absolute sample position, gets the corresponding bar and beat position.*
- virtual [AAX_Result GetTicksPerQuarter](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per quarter note.
 - virtual [AAX_Result GetCurrentTicksPerBeat](#) (uint32_t *ticks) const =0
CALL: Retrieves the number of ticks per beat.
 - virtual [AAX_Result GetTimelineSelectionStartPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the beginning of the current transport selection.
 - virtual [AAX_Result GetTimeCodeInfo](#) (AAX_EFrameRate *oFrameRate, int32_t *oOffset) const =0
CALL: Retrieves the current time code frame rate and offset.
 - virtual [AAX_Result GetFeetFramesInfo](#) (AAX_EFeetFramesRate *oFeetFramesRate, int64_t *oOffset) const =0
CALL: Retrieves the current timecode feet/frames rate and offset.
 - virtual [AAX_Result IsMetronomeEnabled](#) (int32_t *isEnabled) const =0
Sets isEnabled to true if the metronome is enabled.
 - virtual [AAX_Result GetHDTimeCodeInfo](#) (AAX_EFrameRate *oHDFrameRate, int64_t *oHDOffset) const =0
CALL: Retrieves the current HD time code frame rate and offset.
 - virtual [AAX_Result RequestTransportStart](#) ()=0
CALL: Request that the host transport start playback.
 - virtual [AAX_Result RequestTransportStop](#) ()=0
CALL: Request that the host transport stop playback.
 - virtual [AAX_Result GetTimelineSelectionEndPosition](#) (int64_t *oSampleLocation) const =0
CALL: Retrieves the absolute sample position of the end of the current transport selection.

14.161.2 Constructor & Destructor Documentation

14.161.2.1 AAX_VTransport()

```
AAX_VTransport::AAX_VTransport (
    IACFUnknown * pUnknown )
```

14.161.2.2 ~AAX_VTransport()

```
AAX_VTransport::~~AAX_VTransport ( )
```

14.161.3 Member Function Documentation

14.161.3.1 GetCurrentTempo()

```
AAX_Result AAX_VTransport::GetCurrentTempo (
    double * TempoBPM ) const [virtual]
```

CALL: Gets the current tempo.

Returns the tempo corresponding to the current position of the transport counter

Note

The resolution of the tempo returned here is based on the host's tempo resolution, so it will match the tempo displayed in the host. Use [GetCurrentTicksPerBeat\(\)](#) to calculate the tempo resolution note.

Parameters

out	<i>TempoBPM</i>	The current tempo in beats per minute
-----	-----------------	---------------------------------------

Implements [AAX_ITransport](#).

14.161.3.2 GetCurrentMeter()

```
AAX_Result AAX_VTransport::GetCurrentMeter (
    int32_t * MeterNumerator,
    int32_t * MeterDenominator ) const [virtual]
```

CALL: Gets the current meter.

Returns the meter corresponding to the current position of the transport counter

Parameters

out	<i>MeterNumerator</i>	The numerator portion of the meter
out	<i>MeterDenominator</i>	The denominator portion of the meter

Implements [AAX_ITransport](#).

14.161.3.3 IsTransportPlaying()

```
AAX_Result AAX_VTransport::IsTransportPlaying (
    bool * isPlaying ) const [virtual]
```

CALL: Indicates whether or not the transport is playing back.

Parameters

out	<i>isPlaying</i>	true if the transport is currently in playback
-----	------------------	--

Implements [AAX_ITransport](#).

14.161.3.4 GetCurrentTickPosition()

```
AAX_Result AAX_VTransport::GetCurrentTickPosition (
    int64_t * TickPosition ) const [virtual]
```

CALL: Gets the current tick position.

Returns the current tick position corresponding to the current transport position. One "Tick" is represented here as 1/960000 of a quarter note. That is, there are 960,000 of these ticks in a quarter note.

Host Compatibility Notes The tick resolution here is different than that of the tick displayed in Pro Tools. "Display ticks" (as they are called) are 1/960 of a quarter note.

Parameters

out	<i>TickPosition</i>	The tick position value
-----	---------------------	-------------------------

Implements [AAX_ITransport](#).

14.161.3.5 GetCurrentLoopPosition()

```
AAX_Result AAX_VTransport::GetCurrentLoopPosition (
    bool * bLooping,
    int64_t * LoopStartTick,
    int64_t * LoopEndTick ) const [virtual]
```

CALL: Gets current information on loop playback.

Host Compatibility Notes This does not indicate anything about the status of the "Loop Record" option. Even when the host is configured to loop playback, looping may not occur if certain conditions are not met (i.e. the length of the selection is too short)

Parameters

out	<i>bLooping</i>	true if the host is configured to loop playback
out	<i>LoopStartTick</i>	The starting tick position of the selection being looped (see GetCurrentTickPosition())
out	<i>LoopEndTick</i>	The ending tick position of the selection being looped (see GetCurrentTickPosition())

Implements [AAX_ITransport](#).

14.161.3.6 GetCurrentNativeSampleLocation()

```
AAX_Result AAX_VTransport::GetCurrentNativeSampleLocation (
    int64_t * SampleLocation ) const [virtual]
```

CALL: Gets the current playback location of the native audio engine.

When called from a ProcessProc render callback, this method will provide the absolute sample location at the beginning of the callback's audio buffers.

When called from [AAX_IEffectParameters::RenderAudio_Hybrid\(\)](#), this method will provide the absolute sample location for the samples in the method's **output** audio buffers. To calculate the absolute sample location for the samples in the method's input buffers (i.e. the timeline location where the samples originated) subtract the value provided by [AAX_IController::GetHybridSignalLatency\(\)](#) from this value.

When called from a non-real-time thread, this method will provide the current location of the samples being processed by the plug-in's ProcessProc on its real-time processing thread.

Note

This method only returns a value during playback. It cannot be used to determine, e.g., the location of the timeline selector while the host is not in playback.

Parameters

out	<i>SampleLocation</i>	Absolute sample location of the first sample in the current native processing buffer
-----	-----------------------	--

Implements [AAX_ITransport](#).

14.161.3.7 GetCustomTickPosition()

```
AAX_Result AAX_VTransport::GetCustomTickPosition (
    int64_t * oTickPosition,
    int64_t iSampleLocation ) const [virtual]
```

CALL: Given an absolute sample position, gets the corresponding tick position.

Host Compatibility Notes There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Parameters

out	<i>oTickPosition</i>	the timeline tick position corresponding to <i>iSampleLocation</i>
in	<i>iSampleLocation</i>	An absolute sample location (see GetCurrentNativeSampleLocation())

Implements [AAX_ITransport](#).

14.161.3.8 GetBarBeatPosition()

```
AAX_Result AAX_VTransport::GetBarBeatPosition (
    int32_t * Bars,
    int32_t * Beats,
    int64_t * DisplayTicks,
    int64_t SampleLocation ) const [virtual]
```

CALL: Given an absolute sample position, gets the corresponding bar and beat position.

Host Compatibility Notes There is a minor performance cost associated with using this API in Pro Tools. It should not be used excessively without need.

Parameters

out	<i>Bars</i>	The bar corresponding to <code>SampleLocation</code>
out	<i>Beats</i>	The beat corresponding to <code>SampleLocation</code>
out	<i>DisplayTicks</i>	The ticks corresponding to <code>SampleLocation</code>
in	<i>SampleLocation</i>	An absolute sample location (see GetCurrentNativeSampleLocation())

Implements [AAX_ITransport](#).

14.161.3.9 GetTicksPerQuarter()

```
AAX_Result AAX_VTransport::GetTicksPerQuarter (
    uint32_t * ticks ) const [virtual]
```

CALL: Retrieves the number of ticks per quarter note.

Parameters

out	<i>ticks</i>	
-----	--------------	--

Implements [AAX_ITransport](#).

14.161.3.10 GetCurrentTicksPerBeat()

```
AAX_Result AAX_VTransport::GetCurrentTicksPerBeat (
    uint32_t * ticks ) const [virtual]
```

CALL: Retrieves the number of ticks per beat.

Parameters

out	<i>ticks</i>	
-----	--------------	--

Implements [AAX_ITransport](#).

14.161.3.11 GetTimelineSelectionStartPosition()

```
AAX_Result AAX_VTransport::GetTimelineSelectionStartPosition (
    int64_t * oSampleLocation ) const [virtual]
```

CALL: Retrieves the absolute sample position of the beginning of the current transport selection.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oSampleLocation</i>	
-----	------------------------	--

Implements [AAX_ITransport](#).

14.161.3.12 GetTimeCodeInfo()

```
AAX_Result AAX_VTransport::GetTimeCodeInfo (
    AAX_EFrameRate * oFrameRate,
    int32_t * oOffset ) const [virtual]
```

CALL: Retrieves the current time code frame rate and offset.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oFrameRate</i>	
out	<i>oOffset</i>	

Implements [AAX_ITransport](#).

14.161.3.13 GetFeetFramesInfo()

```
AAX_Result AAX_VTransport::GetFeetFramesInfo (
    AAX_EFeetFramesRate * oFeetFramesRate,
    int64_t * oOffset ) const [virtual]
```

CALL: Retrieves the current timecode feet/frames rate and offset.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>oFeetFramesRate</i>	
out	<i>oOffset</i>	

Implements [AAX_ITransport](#).

14.161.3.14 IsMetronomeEnabled()

```
AAX_Result AAX_VTransport::IsMetronomeEnabled (
    int32_t * isEnabled ) const [virtual]
```

Sets isEnabled to true if the metronome is enabled.

Note

This method is part of the [version 2 transport interface](#)

Parameters

out	<i>isEnabled</i>	
-----	------------------	--

Implements [AAX_ITransport](#).

14.161.3.15 GetHDTimeCodeInfo()

```
AAX_Result AAX_VTransport::GetHDTimeCodeInfo (
    AAX_EFrameRate * oHDFrameRate,
    int64_t * oHDOffset ) const [virtual]
```

CALL: Retrieves the current HD time code frame rate and offset.

Note

This method is part of the [version 3 transport interface](#)

Parameters

out	<i>oHDFrameRate</i>	
out	<i>oHDOffset</i>	

Implements [AAX_ITransport](#).

14.161.3.16 GetTimelineSelectionEndPosition()

```
AAX_Result AAX_VTransport::GetTimelineSelectionEndPosition (
    int64_t * oSampleLocation ) const [virtual]
```

CALL: Retrieves the absolute sample position of the end of the current transport selection.

Note

This method is part of the [version 4 transport interface](#)

Parameters

out	<i>oSampleLocation</i>	
-----	------------------------	--

Implements [AAX_ITransport](#).

14.161.3.17 RequestTransportStart()

```
AAX_Result AAX_VTransport::RequestTransportStart ( ) [virtual]
```

CALL: Request that the host transport start playback.

Note

This method is part of the [AAX_IACFTransportControl](#) interface

Implements [AAX_ITransport](#).

14.161.3.18 RequestTransportStop()

```
AAX_Result AAX_VTransport::RequestTransportStop ( ) [virtual]
```

CALL: Request that the host transport stop playback.

Note

This method is part of the [AAX_IACFTransportControl](#) interface

Implements [AAX_ITransport](#).

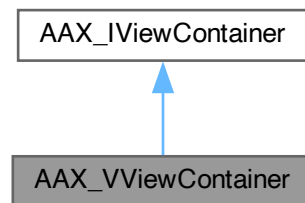
The documentation for this class was generated from the following file:

- [AAX_VTransport.h](#)

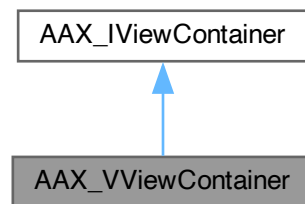
14.162 AAX_VViewContainer Class Reference

```
#include <AAX_VViewContainer.h>
```

Inheritance diagram for AAX_VViewContainer:



Collaboration diagram for AAX_VViewContainer:



14.162.1 Description

Version-managed concrete [AAX_IViewContainer](#) class.

Public Member Functions

- [AAX_VViewContainer](#) ([IACFUnknown](#) *pUnknown)
- [~AAX_VViewContainer](#) () [AAX_OVERRIDE](#)
- [int32_t](#) [GetType](#) () [AAX_OVERRIDE](#)
Returns the raw view type as one of [AAX_EViewContainer_Type](#).
- [void](#) * [GetPtr](#) () [AAX_OVERRIDE](#)
Returns a pointer to the raw view.
- [AAX_Result](#) [GetModifiers](#) ([uint32_t](#) *outModifiers) [AAX_OVERRIDE](#)
Queries the host for the current [modifier keys](#).

- [AAX_Result SetViewSize](#) ([AAX_Point](#) &inSize) [AAX_OVERRIDE](#)
Request a change to the main view size.
- [AAX_Result HandleParameterMouseDown](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse down event.
- [AAX_Result HandleParameterMouseDrag](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse drag event.
- [AAX_Result HandleParameterMouseUp](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse up event.
- [AAX_Result HandleParameterMouseEnter](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse enter event to the parameter's control.
- [AAX_Result HandleParameterMouseExit](#) ([AAX_CParamID](#) inParamID, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse exit event from the parameter's control.
- [AAX_Result HandleMultipleParametersMouseDown](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse down event.
- [AAX_Result HandleMultipleParametersMouseDrag](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse drag event.
- [AAX_Result HandleMultipleParametersMouseUp](#) (const [AAX_CParamID](#) *inParamIDs, uint32_t inNumOfParams, uint32_t inModifiers) [AAX_OVERRIDE](#)
Alert the host to a mouse up event.

Public Member Functions inherited from [AAX_IViewContainer](#)

- virtual [~AAX_IViewContainer](#) (void)

14.162.2 Constructor & Destructor Documentation

14.162.2.1 [AAX_VViewContainer\(\)](#)

```
AAX_VViewContainer::AAX_VViewContainer (
    IACFUnknown * pUnknown )
```

14.162.2.2 [~AAX_VViewContainer\(\)](#)

```
AAX_VViewContainer::~~AAX_VViewContainer ( )
```

14.162.3 Member Function Documentation

14.162.3.1 GetType()

```
int32_t AAX_VViewContainer::GetType ( ) [virtual]
```

Returns the raw view type as one of [AAX_EViewContainer_Type](#).

Implements [AAX_IViewContainer](#).

14.162.3.2 GetPtr()

```
void * AAX_VViewContainer::GetPtr ( ) [virtual]
```

Returns a pointer to the raw view.

Implements [AAX_IViewContainer](#).

14.162.3.3 GetModifiers()

```
AAX_Result AAX_VViewContainer::GetModifiers (
    uint32_t * outModifiers ) [virtual]
```

Queries the host for the current [modifier keys](#).

This method returns a bit mask with bits set for each of the currently active modifier keys. This method does not return the state of the [AAX_eModifiers_SecondaryButton](#).

Host Compatibility Notes Although this method allows plug-ins to acquire the current state of the Windows key (normally blocked by Pro Tools), plug-ins should not use key combinations that require this key.

Parameters

out	<i>outModifiers</i>	Current modifiers as a bitmask of AAX_EModifiers
-----	---------------------	--

Implements [AAX_IViewContainer](#).

14.162.3.4 SetViewSize()

```
AAX_Result AAX_VViewContainer::SetViewSize (
    AAX_Point & inSize ) [virtual]
```

Request a change to the main view size.

Note

- For compatibility with the smallest supported displays, plug-in GUI dimensions should not exceed 749x617 pixels, or 749x565 pixels for plug-ins with sidechain support.

Parameters

in	<i>inSize</i>	The new size to which the plug-in view should be set
----	---------------	--

Implements [AAX_IViewContainer](#).

14.162.3.5 HandleParameterMouseDown()

```
AAX_Result AAX_VViewContainer::HandleParameterMouseDown (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse down event.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implements [AAX_IViewContainer](#).

14.162.3.6 HandleParameterMouseDrag()

```
AAX_Result AAX_VViewContainer::HandleParameterMouseDrag (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse drag event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implements [AAX_IViewContainer](#).

14.162.3.7 HandleParameterMouseUp()

```
AAX_Result AAX_VViewContainer::HandleParameterMouseUp (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse up event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being edited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implements [AAX_IViewContainer](#).

14.162.3.8 HandleParameterMouseEnter()

```
AAX_Result AAX_VViewContainer::HandleParameterMouseEnter (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse enter event to the parameter's control.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being entered
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Returns [AAX_SUCCESS](#) if event was processed successfully, otherwise an [AAX_ERROR](#) code

Implements [AAX_IViewContainer](#).

14.162.3.9 HandleParameterMouseExit()

```
AAX_Result AAX_VViewContainer::HandleParameterMouseExit (
    AAX_CParamID inParamID,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse exit event from the parameter's control.

Parameters

in	<i>inParamID</i>	ID of the parameter whose control is being exited
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Returns [AAX_SUCCESS](#) if event was processed successfully, otherwise an [AAX_ERROR](#) code

Implements [AAX_IViewContainer](#).

14.162.3.10 HandleMultipleParametersMouseDown()

```
AAX\_Result AAX_VViewContainer::HandleMultipleParametersMouseDown (
    const AAX\_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse down event.

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implements [AAX_IViewContainer](#).

14.162.3.11 HandleMultipleParametersMouseDrag()

```
AAX\_Result AAX_VViewContainer::HandleMultipleParametersMouseDrag (
    const AAX\_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse drag event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implements [AAX_IViewContainer](#).

14.162.3.12 HandleMultipleParametersMouseUp()

```
AAX_Result AAX_VViewContainer::HandleMultipleParametersMouseUp (
    const AAX_CParamID * inParamIDs,
    uint32_t inNumOfParams,
    uint32_t inModifiers ) [virtual]
```

Alert the host to a mouse up event.

Warning

The host may return [AAX_ERROR_UNIMPLEMENTED](#) for this event even if the host did handle the corresponding mouse down event. A plug-in should ignore any following mouse drag and mouse up events that correspond to a host-managed mouse down event. ([PTSW-195209](#) / [PT-218474](#))

Parameters

in	<i>inParamIDs</i>	IDs of the parameters that belong to the same GUI element whose controls are being edited
in	<i>inNumOfParams</i>	Number of parameter IDS
in	<i>inModifiers</i>	A bitmask of AAX_EModifiers values

Implements [AAX_IViewContainer](#).

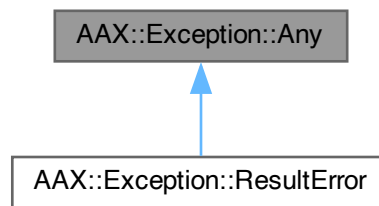
The documentation for this class was generated from the following file:

- [AAX_VViewContainer.h](#)

14.163 AAX::Exception::Any Class Reference

```
#include <AAX_Exception.h>
```

Inheritance diagram for AAX::Exception::Any:



14.163.1 Description

Base class for AAX exceptions

This class is defined within the AAX Library and is always handled within the AAX plug-in. Objects of this class are never passed between the plug-in and the AAX host.

The definition of this class may change between versions of the AAX SDK. This class does not include any form of version safety for cross-version compatibility.

Warning

- Do not use multiple inheritance in any sub-classes within the [AAX::Exception::Any](#) inheritance tree
- Never pass exceptions across the library boundary to the AAX host

Public Member Functions

- virtual [~Any](#) ()
- template<class C >
 [Any](#) (const C &inWhat)
- template<class C1 , class C2 , class C3 >
 [Any](#) (const C1 &inWhat, const C2 &inFunction, const C3 &inLine)
- [Any](#) & [operator=](#) (const [Any](#) &inOther)
- [AAX_DEFAULT_MOVE_CTOR](#) ([Any](#))
- [AAX_DEFAULT_MOVE_OPER](#) ([Any](#))
- const std::string & [What](#) () const
- const std::string & [Desc](#) () const
- const std::string & [Function](#) () const
- const std::string & [Line](#) () const

14.163.2 Constructor & Destructor Documentation

14.163.2.1 ~Any()

```
virtual AAX::Exception::Any::~Any ( ) [inline], [virtual]
```

14.163.2.2 Any() [1/2]

```
template<class C >
AAX::Exception::Any::Any (
    const C & inWhat ) [inline], [explicit]
```

Explicit conversion from a string-like object

14.163.2.3 Any() [2/2]

```
template<class C1 , class C2 , class C3 >
AAX::Exception::Any::Any (
    const C1 & inWhat,
    const C2 & inFunction,
    const C3 & inLine ) [inline], [explicit]
```

Explicit conversion from a string-like object with function name and line number

14.163.3 Member Function Documentation

14.163.3.1 operator=()

```
Any & AAX::Exception::Any::operator= (
    const Any & inOther ) [inline]
```

14.163.3.2 AAX_DEFAULT_MOVE_CTOR()

```
AAX::Exception::Any::AAX_DEFAULT_MOVE_CTOR (
    Any )
```

14.163.3.3 AAX_DEFAULT_MOVE_OPER()

```
AAX::Exception::Any::AAX_DEFAULT_MOVE_OPER (
    Any )
```

14.163.3.4 What()

```
const std::string & AAX::Exception::Any::What ( ) const [inline]
```

Referenced by [AAX::AsString\(\)](#).

Here is the caller graph for this function:



14.163.3.5 Desc()

```
const std::string & AAX::Exception::Any::Desc ( ) const [inline]
```

14.163.3.6 Function()

```
const std::string & AAX::Exception::Any::Function ( ) const [inline]
```

14.163.3.7 Line()

```
const std::string & AAX::Exception::Any::Line ( ) const [inline]
```

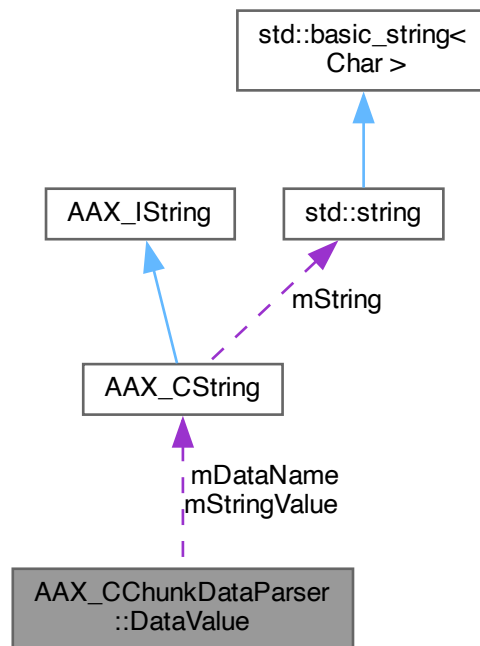
The documentation for this class was generated from the following file:

- [AAX_Exception.h](#)

14.164 AAX_CChunkDataParser::DataValue Struct Reference

```
#include <AAX_CChunkDataParser.h>
```

Collaboration diagram for AAX_CChunkDataParser::DataValue:



Public Member Functions

- [DataValue\(\)](#)

Public Attributes

- [int32_t mDataType](#)
- [AAX_CString mDataName](#)
name of the stored data
- [int64_t mIntValue](#)
used if this [DataValue](#) is not a string
- [AAX_CString mStringValue](#)
used if this [DataValue](#) is a string

14.164.1 Constructor & Destructor Documentation

14.164.1.1 `DataValue()`

```
AAX_CChunkDataParser::DataValue::DataValue ( ) [inline]
```

14.164.2 Member Data Documentation

14.164.2.1 `mDataType`

```
int32_t AAX_CChunkDataParser::DataValue::mDataType
```

14.164.2.2 `mDataName`

```
AAX\_CString AAX_CChunkDataParser::DataValue::mDataName
```

name of the stored data

14.164.2.3 `mIntValue`

```
int64_t AAX_CChunkDataParser::DataValue::mIntValue
```

used if this [DataValue](#) is not a string

14.164.2.4 `mStringValue`

```
AAX\_CString AAX_CChunkDataParser::DataValue::mStringValue
```

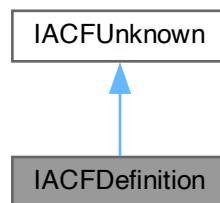
used if this [DataValue](#) is a string

The documentation for this struct was generated from the following file:

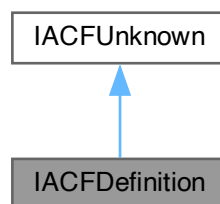
- [AAX_CChunkDataParser.h](#)

14.165 IACFDefinition Interface Reference

Inheritance diagram for IACFDefinition:



Collaboration diagram for IACFDefinition:



14.165.1 Description

Publicly inherits from IACFUnknown. This abstract interface is used to identify all of the plug-in components in the host.

Remarks

This interface is the base class for both plug-in and component definitions. All defined attributes are read only.

Note

This interface does not provide any attribute enumeration. You must know the uid of the associated with the attribute that you need to find.

This interface is implemented by the host. The plug-in will use this interface to define optional attributes for both plug-in and component implementations classes.

Public Member Functions

- virtual ACFRESULT ACFMETHODCALLTYPE [DefineAttribute](#) (const [acfUID](#) &attributeID, const [acfUID](#) &typeID, const void *attrData, acfUInt32 attrDataSize)=0
Add a read only attribute to the definition.
- virtual ACFRESULT ACFMETHODCALLTYPE [GetAttributeInfo](#) (const [acfUID](#) &attributeID, [acfUID](#) *typeID, acfUInt32 *attrDataSize)=0
Returns information about the given attribute.
- virtual ACFRESULT ACFMETHODCALLTYPE [CopyAttribute](#) (const [acfUID](#) &attributeID, const [acfUID](#) &typeID, void *attrData, acfUInt32 attrDataSize)=0
Copy the a given attribute.

Public Member Functions inherited from [IACFUnknown](#)

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppvOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.165.2 Member Function Documentation

14.165.2.1 DefineAttribute()

```
virtual ACFRESULT ACFMETHODCALLTYPE IACFDefinition::DefineAttribute (
    const acfUID & attributeID,
    const acfUID & typeID,
    const void * attrData,
    acfUInt32 attrDataSize ) [pure virtual]
```

Add a read only attribute to the definition.

DefineAttribute

Remarks

Use the method to define additional global attributes for you component. This method will fail if the attribute has already been defined.

Parameters

<i>attributeID</i>	Unique identifier for attribute
<i>typeID</i>	Indicates the type of the attribute data
<i>attrData</i>	Pointer to buffer that contains the attribute data
<i>attrDataSize</i>	Size of the attribute buffer

14.165.2.2 GetAttributeInfo()

```
virtual ACFRESULT ACFMETHODCALLTYPE IACFDefinition::GetAttributeInfo (
    const acfUID & attributeID,
    acfUID * typeID,
    acfUInt32 * attrDataSize ) [pure virtual]
```

Returns information about the given attribute.

Remarks

Use this method to retrieve the type and size of a given attribute.

Parameters

<i>attributeID</i>	Unique identifier for attribute
<i>typeID</i>	Indicates the type of the attribute data
<i>attrDataSize</i>	Size of the attribute data

14.165.2.3 CopyAttribute()

```
virtual ACFRESULT ACFMETHODCALLTYPE IACFDefinition::CopyAttribute (
    const acfUID & attributeID,
    const acfUID & typeID,
    void * attrData,
    acfUInt32 attrDataSize ) [pure virtual]
```

Copy the a given attribute.

CopyAttribute

Remarks

Use this method to access the contents of a given attribute.

Parameters

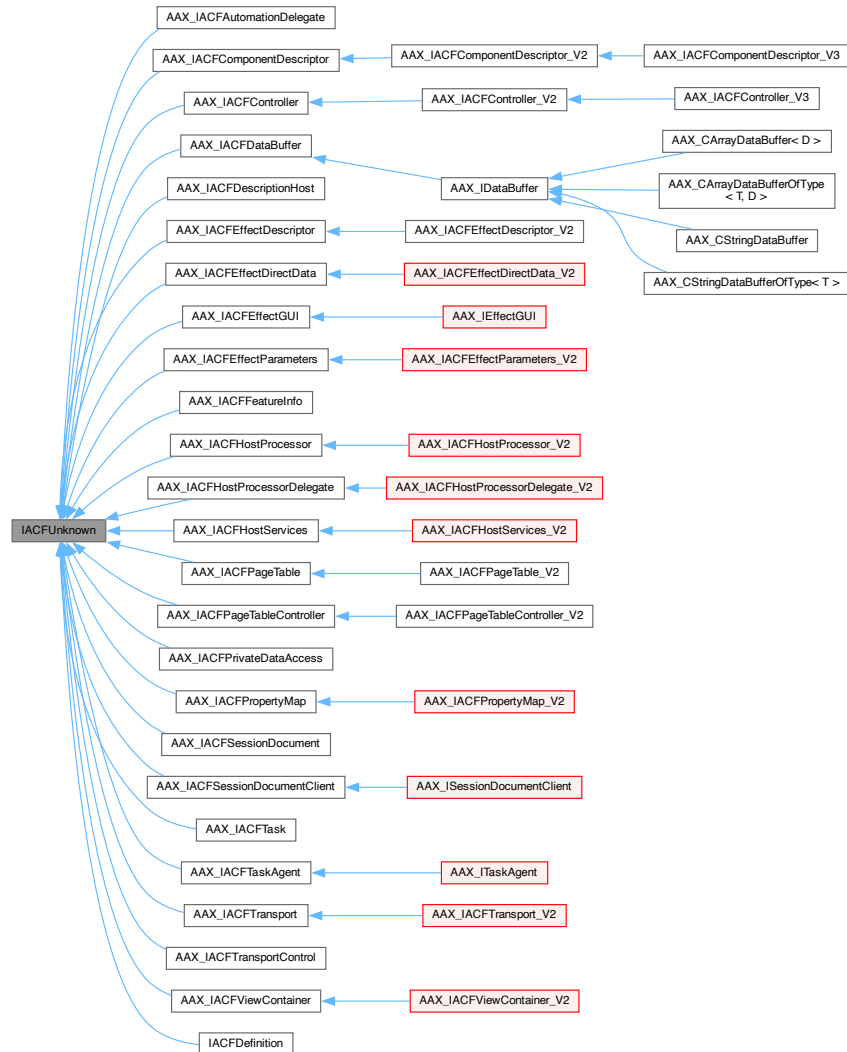
<i>attributeID</i>	Unique identifier for attribute
<i>typeID</i>	Indicates the type of the attribute data
<i>attrData</i>	Pointer to buffer to copy the attribute data
<i>attrDataSize</i>	Size of the attribute buffer

The documentation for this interface was generated from the following file:

- [AAX_ACFInterface.doxygen](#)

14.166 IACFUnknown Interface Reference

Inheritance diagram for IACFUnknown:



14.166.1 Description

COM compatible IUnknown C++ interface.

Remarks

The methods of the [IACFUnknown](#) interface, implemented by all ACF objects, supports general inter-object protocol negotiation via the `QueryInterface` method, and object lifetime management with the `AddRef` and `Release` methods.

Note

Because AddRef and Release are not required to return accurate values, callers of these methods must not use the return values to determine if an object is still valid or has been destroyed. (Standard M*cr*s*ft disclaimer)

For further information please refer to the Microsoft documentation for IUnknown.

Note

This class will work only with compilers that can produce COM-compatible object layouts for C++ classes. egcs can not do this. Metrowerks can do this (if you subclass from __comobject).

Public Member Functions

- virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE [QueryInterface](#) (const [acfIID](#) &iid, void **ppOut)=0
Returns pointers to supported interfaces.
- virtual acfUInt32 ACFMETHODCALLTYPE [AddRef](#) (void)=0
Increments reference count.
- virtual acfUInt32 ACFMETHODCALLTYPE [Release](#) (void)=0
Decrements reference count.

14.166.2 Member Function Documentation**14.166.2.1 QueryInterface()**

```
virtual BEGIN_ACFINTERFACE ACFRESULT ACFMETHODCALLTYPE IACFUnknown::QueryInterface (
    const acfIID & iid,
    void ** ppOut ) [pure virtual]
```

Returns pointers to supported interfaces.

Remarks

The QueryInterface method gives a client access to alternate interfaces implemented by an object. The returned interface pointer will have already had its reference count incremented so the caller will be required to call the Release method.

Parameters

<i>iid</i>	Identifier of the requested interface
<i>ppOut</i>	Address of variable that receives the interface pointer associated with iid.

14.166.2.2 AddRef()

```
virtual acfUInt32 ACFMETHODCALLTYPE IACFUnknown::AddRef (
    void ) [pure virtual]
```

Increments reference count.

Remarks

The AddRef method should be called every time a new copy of an interface is made. When this copy is no longer referenced it must be released with the Release method.

14.166.2.3 Release()

```
virtual acfUInt32 ACFMETHODCALLTYPE IACFUnknown::Release (
    void ) [pure virtual]
```

Decrements reference count.

Remarks

Use this method to decrement the reference count. When the reference count reaches zero the object that implements the interface will be deleted.

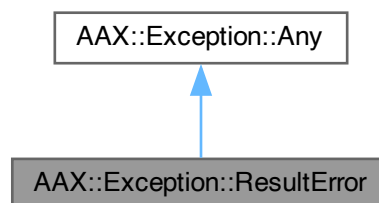
The documentation for this interface was generated from the following file:

- [AAX_ACFInterface.doxygen](#)

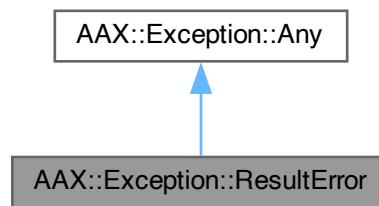
14.167 AAX::Exception::ResultError Class Reference

```
#include <AAX_Exception.h>
```

Inheritance diagram for AAX::Exception::ResultError:



Collaboration diagram for AAX::Exception::ResultError:



14.167.1 Description

[Exception](#) class for [AAX_EError](#) results

Public Member Functions

- [ResultError](#) ([AAX_Result](#) inWhatResult)
- template<class C >
 [ResultError](#) ([AAX_Result](#) inWhatResult, const C &inFunction)
- template<class C1 , class C2 >
 [ResultError](#) ([AAX_Result](#) inWhatResult, const C1 &inFunction, const C2 &inLine)
- [AAX_Result Result](#) () const

Public Member Functions inherited from [AAX::Exception::Any](#)

- virtual [~Any](#) ()
- template<class C >
 [Any](#) (const C &inWhat)
- template<class C1 , class C2 , class C3 >
 [Any](#) (const C1 &inWhat, const C2 &inFunction, const C3 &inLine)
- [Any](#) & [operator=](#) (const [Any](#) &inOther)
- [AAX_DEFAULT_MOVE_CTOR](#) ([Any](#))
- [AAX_DEFAULT_MOVE_OPER](#) ([Any](#))
- const std::string & [What](#) () const
- const std::string & [Desc](#) () const
- const std::string & [Function](#) () const
- const std::string & [Line](#) () const

Static Public Member Functions

- static std::string [FormatResult](#) ([AAX_Result](#) inResult)

14.167.2 Constructor & Destructor Documentation

14.167.2.1 ResultError() [1/3]

```
AAX::Exception::ResultError::ResultError (
    AAX_Result inWhatResult ) [inline], [explicit]
```

14.167.2.2 ResultError() [2/3]

```
template<class C >
AAX::Exception::ResultError::ResultError (
    AAX_Result inWhatResult,
    const C & inFunction ) [inline], [explicit]
```

14.167.2.3 ResultError() [3/3]

```
template<class C1 , class C2 >
AAX::Exception::ResultError::ResultError (
    AAX_Result inWhatResult,
    const C1 & inFunction,
    const C2 & inLine ) [inline], [explicit]
```

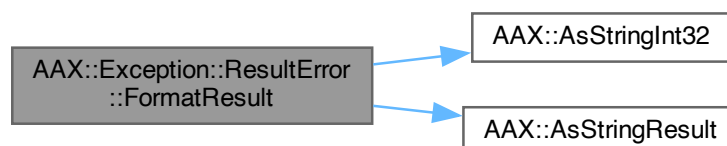
14.167.3 Member Function Documentation

14.167.3.1 FormatResult()

```
static std::string AAX::Exception::ResultError::FormatResult (
    AAX_Result inResult ) [inline], [static]
```

References [AAX::AsStringInt32\(\)](#), and [AAX::AsStringResult\(\)](#).

Here is the call graph for this function:



14.167.3.2 Result()

```
AAX_Result AAX::Exception::ResultError::Result ( ) const [inline]
```

The documentation for this class was generated from the following file:

- [AAX_Exception.h](#)

14.168 SAutoArray< T > Struct Template Reference**Public Member Functions**

- [SAutoArray](#) ()
- [~SAutoArray](#) ()
- void [Reset](#) (T *inData)
- T * [Get](#) ()

14.168.1 Constructor & Destructor Documentation**14.168.1.1 SAutoArray()**

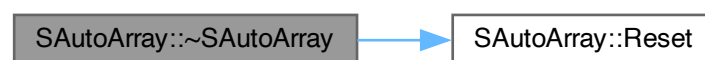
```
template<typename T >
SAutoArray< T >::SAutoArray ( ) [inline]
```

14.168.1.2 ~SAutoArray()

```
template<typename T >
SAutoArray< T >::~~SAutoArray ( ) [inline]
```

References [SAutoArray< T >::Reset\(\)](#).

Here is the call graph for this function:



14.168.2 Member Function Documentation

14.168.2.1 Reset()

```
template<typename T >
void SAutoArray< T >::Reset (
    T * inData ) [inline]
```

Referenced by [SAutoArray< T >::~~SAutoArray\(\)](#).

Here is the caller graph for this function:



14.168.2.2 Get()

```
template<typename T >
T * SAutoArray< T >::Get ( ) [inline]
```

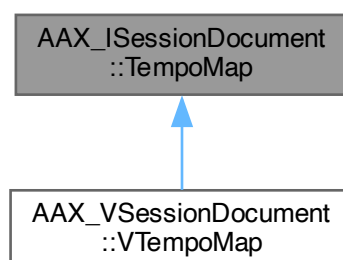
The documentation for this struct was generated from the following file:

- [AAX_MIDILogging.cpp](#)

14.169 AAX_I_SessionDocument::TempoMap Class Reference

```
#include <AAX_I_SessionDocument.h>
```

Inheritance diagram for `AAX_I_SessionDocument::TempoMap`:



Public Member Functions

- virtual [~TempoMap](#) ()=default
- virtual int32_t [Size](#) () const =0
- virtual [AAX_CTempoBreakpoint](#) const * [Data](#) () const =0

14.169.1 Constructor & Destructor Documentation

14.169.1.1 ~TempoMap()

```
virtual AAX_I_SessionDocument::TempoMap::~~TempoMap ( ) [virtual], [default]
```

14.169.2 Member Function Documentation

14.169.2.1 Size()

```
virtual int32_t AAX_I_SessionDocument::TempoMap::Size ( ) const [pure virtual]
```

Implemented in [AAX_V_SessionDocument::VTempoMap](#).

14.169.2.2 Data()

```
virtual AAX\_CTempoBreakpoint const * AAX_I_SessionDocument::TempoMap::Data ( ) const [pure virtual]
```

Implemented in [AAX_V_SessionDocument::VTempoMap](#).

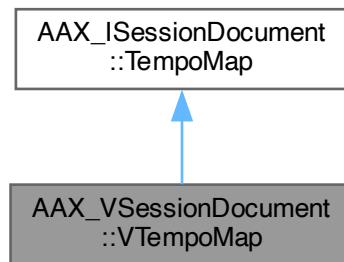
The documentation for this class was generated from the following file:

- [AAX_I_SessionDocument.h](#)

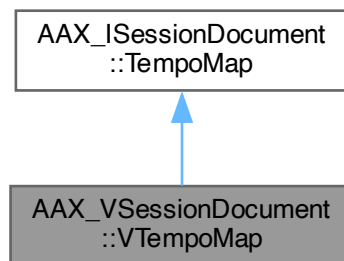
14.170 AAX_VSessionDocument::VTempoMap Class Reference

```
#include <AAX_VSessionDocument.h>
```

Inheritance diagram for AAX_VSessionDocument::VTempoMap:



Collaboration diagram for AAX_VSessionDocument::VTempoMap:



Public Member Functions

- [~VTempoMap\(\)](#) [AAX_OVERRIDE](#)
- [VTempoMap\(IACFUnknown &inDataBuffer\)](#)
- [int32_t Size\(\)](#) const [AAX_OVERRIDE](#)
- [AAX_CTempoBreakpoint](#) const * [Data\(\)](#) const [AAX_OVERRIDE](#)

Public Member Functions inherited from [AAX_ISessionDocument::TempoMap](#)

- virtual [~TempoMap\(\)](#)=default
- virtual [int32_t Size\(\)](#) const =0
- virtual [AAX_CTempoBreakpoint](#) const * [Data\(\)](#) const =0

14.170.1 Constructor & Destructor Documentation

14.170.1.1 ~VTempoMap()

AAX_VSessionDocument::VTempoMap::~~VTempoMap ()

14.170.1.2 VTempoMap()

AAX_VSessionDocument::VTempoMap::VTempoMap (
 [IACFUnknown](#) & *inDataBuffer*) [explicit]

14.170.2 Member Function Documentation

14.170.2.1 Size()

int32_t AAX_VSessionDocument::VTempoMap::Size () const [virtual]

Implements [AAX_ISessionDocument::TempoMap](#).

14.170.2.2 Data()

[AAX_CTempoBreakpoint](#) const * AAX_VSessionDocument::VTempoMap::Data () const [virtual]

Implements [AAX_ISessionDocument::TempoMap](#).

The documentation for this class was generated from the following file:

- [AAX_VSessionDocument.h](#)

Chapter 15

File Documentation

15.1 AAX_ACFInterface.doxygen File Reference

Classes

- struct [_acfUID](#)
- interface [IACFUnknown](#)
COM compatible IUnknown C++ interface.
- interface [IACFDefinition](#)
Publicly inherits from IACFUnknown. This abstract interface is used to indentify all of the plug-in components in the host.

Typedefs

- typedef struct [_acfUID](#) [acfUID](#)
GUID compatible structure for ACF.
- typedef [acfUID](#) [acfIID](#)
IID compatible structure for ACF.

15.1.1 Typedef Documentation

15.1.1.1 acfUID

[acfUID](#)

GUID compatible structure for ACF.

15.1.1.2 acfIID

[acfIID](#)

IID compatible structure for ACF.

15.2 AAX_AdditionalFeatures_Algorithm.doxygen File Reference

15.3 AAX_AdditionalFeatures_AOSandSidechain.doxygen File Reference

15.4 AAX_AdditionalFeatures_CurveDisplays.doxygen File Reference

15.5 AAX_AdditionalFeatures_Hybrid.doxygen File Reference

15.6 AAX_AdditionalFeatures_Meters.doxygen File Reference

15.7 AAX_AdditionalFeatures_MIDI.doxygen File Reference

15.8 AAX_AuxInterface_DirectData.doxygen File Reference

15.9 AAX_AuxInterface_HostProcessor.doxygen File Reference

15.10 AAX_AuxInterface_TaskAgent.doxygen File Reference

15.11 AAX_BugList.doxygen File Reference

15.12 AAX_CommonInterface_Algorithm.doxygen File Reference

15.13 AAX_CommonInterface_Communication.doxygen File Reference

15.14 AAX_CommonInterface_DataModel.doxygen File Reference

15.15 AAX_CommonInterface_Describe.doxygen File Reference

Functions

- [AAX_Result GetEffectDescriptions](#) ([AAX_ICollection](#) *inCollection)
The plug-in's static Description endpoint.

15.16 AAX_CommonInterface_FormatSpecification.doxygen File Reference

15.17 AAX_CommonInterface_GUI.doxygen File Reference

15.18 AAX_DigiTrace_Guide.doxygen File Reference

15.19 AAX_DistributingYourPlugIn.doxygen File Reference

15.20 AAX_DocsDirectory.doxygen File Reference

15.21 AAX_Getting_Started_Guide.doxygen File Reference

15.22 AAX_HostSupport.doxygen File Reference

15.23 AAX_InstrumentParameters.doxygen File Reference

15.24 AAX_InterfaceList.doxygen File Reference

15.25 AAX_LinkedParameters.doxygen File Reference

15.26 AAX_Media_Composer_Guide.doxygen File Reference

15.27 AAX_OtherExtensions.doxygen File Reference

15.28 AAX_Page_Table_Guide.doxygen File Reference

15.29 AAX_ParameterAutomation.doxygen File Reference

15.30 AAX_ParameterManager.doxygen File Reference

15.31 AAX_ParameterUpdateProtocol.doxygen File Reference

15.32 AAX_ParameterUpdateTiming.doxygen File Reference

15.33 AAX_Pro_Tools_Guide.doxygen File Reference

15.34 AAX_RealTimePerformance.doxygen File Reference

15.35 AAX_RelatedTypes.doxygen File Reference

15.36 AAX_SDK_ChangeLog.doxygen File Reference

15.37 AAX_SDK_ExamplePlugIns.doxygen File Reference

15.38 AAX_SDK_GUIExtensions.doxygen File Reference

15.39 AAX_TI_Guide.doxygen File Reference

15.40 AAX_Troubleshooting.doxygen File Reference

15.41 AAX_VENUE_Guide.doxygen File Reference

15.42 DSH_Guide.doxygen File Reference

15.43 DTT_Guide.doxygen File Reference

15.44 ReadMe.doxygen File Reference

15.45 AAX_MIDILogging.cpp File Reference

```
#include "AAX_MIDILogging.h"  
#include "AAX_CString.h"  
#include "AAX_Assert.h"  
#include <map>  
#include <vector>  
#include <algorithm>
```

Classes

- struct [SAutoArray< T >](#)
- class [AAX_IMIDIMessageInfoDelegate](#)

15.46 AAX_MIDILogging.h File Reference

```
#include "AAX.h"
```

15.46.1 Description

Utilities for logging MIDI data.

Namespaces

- namespace [AAX](#)

Functions

MIDI logging utilities

- void [AAX::AsStringMIDIStream_Debug](#) (const [AAX_CMidiStream](#) &inStream, char *outBuffer, int32_t inBuffer↵
BufferSize)

15.47 AAX_MIDILogging.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00016 /*=====*/
00017
00019 #ifndef AAX_MIDILOGGING_H
00020 #define AAX_MIDILOGGING_H
00022
00023 // AAX Includes
00024 #include "AAX.h"
00025
00026 namespace AAX
00027 {
00038     void AsStringMIDIStream_Debug(const AAX_CMidiStream& inStream, char* outBuffer, int32_t
inBuffer↵
00040 }
00041
00043 #endif // AAX_MIDILOGGING_H
```

15.48 AAX_PluginBundleLocation.h File Reference

15.48.1 Description

Utilities for interacting with the host OS.

Namespaces

- namespace [AAX](#)

Functions

Filesystem utilities

- bool [AAX::GetPathToPlugInBundle](#) (const char *iBundleName, int iMaxLength, char *oModuleName)
Retrieve the file path of the .aaxplugin bundle.

15.49 AAX_PlugInBundleLocation.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00016 /*=====*/
00018 #ifndef AAX_PLUGINBUNDLELOCATION_H
00019 #define AAX_PLUGINBUNDLELOCATION_H
00021
00022 namespace AAX
00023 {
00039     bool GetPathToPlugInBundle (const char* iBundleName, int iMaxLength, char* oModuleName);
00041 }
00042
00044 #endif
```

15.50 AAX_CMonolithicParameters.cpp File Reference

```
#include "AAX_CMonolithicParameters.h"
#include "AAX_Exception.h"
```

15.51 AAX_CMonolithicParameters.h File Reference

```
#include "AAX_CEffectParameters.h"
#include "AAX_IEffectDescriptor.h"
#include "AAX_IComponentDescriptor.h"
#include "AAX_IPropertyMap.h"
#include "AAX_CAtomicQueue.h"
#include "AAX_IParameter.h"
#include "AAX_IMIDINode.h"
#include "AAX_IString.h"
#include <set>
#include <list>
#include <utility>
```

15.51.1 Description

A convenience class extending [AAX_CEffectParameters](#) for monolithic instruments.

Classes

- struct [AAX_SInstrumentSetupInfo](#)
Information used to describe the instrument.
- struct [AAX_SInstrumentPrivateData](#)
Utility struct for [AAX_CMonolithicParameters](#).
- struct [AAX_SInstrumentRenderInfo](#)
Information used to parameterize [AAX_CMonolithicParameters::RenderAudio\(\)](#)
- class [AAX_CMonolithicParameters](#)
Extension of the [AAX_CEffectParameters](#) class for monolithic VIs and effects.

Macros

- `#define kMaxAdditionalMIDINodes 15`
- `#define kMaxAuxOutputStems 32`
- `#define kSynchronizedParameterQueueSize 32`

15.51.2 Macro Definition Documentation

15.51.2.1 kMaxAdditionalMIDINodes

```
#define kMaxAdditionalMIDINodes 15
```

15.51.2.2 kMaxAuxOutputStems

```
#define kMaxAuxOutputStems 32
```

15.51.2.3 kSynchronizedParameterQueueSize

```
#define kSynchronizedParameterQueueSize 32
```

15.52 AAX_CMonolithicParameters.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2014-2016, 2019, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00018
00019
00020 #ifndef AAX_CMONOLITHICPARAMETERS_H
00021 #define AAX_CMONOLITHICPARAMETERS_H
00022
00023 // AAX headers
00024 //
00025 // Parent class
00026 #include "AAX_CEffectParameters.h"
00027 //
00028 // Describe
00029 #include "AAX_IEffectDescriptor.h"
00030 #include "AAX_IComponentDescriptor.h"
00031 #include "AAX_IPropertyMap.h"
00032 //
00033 // Utilities
00034 #include "AAX_CAtomicQueue.h"
00035 #include "AAX_IParameter.h"
00036 #include "AAX_IMIDINode.h"
00037 #include "AAX_IString.h"
00038
00039 // Standard Includes
00040 #include <set>
00041 #include <list>
00042 #include <utility>
00043
00044
00045 // Max number of additional midi nodes is 15, for a grand total of 16 input midi nodes. We're not
00046 // aware of any plug-in that uses more.
00047 #define kMaxAdditionalMIDINodes 15
00048
00049 // You can increase this if you need, it is not a system limitation.
00050 #define kMaxAuxOutputStems 32
00051
00052 // You can increase this if you need; it should be set to a value greater than or equal to the number
00053 // of synchronized parameters in your plug-in
00054 #define kSynchronizedParameterQueueSize 32
00055
00060 struct AAX_SInstrumentSetupInfo
00061 {
00062     //Global MIDI Nodes
00063     bool mNeedsGlobalMIDI;
00064     const char* mGlobalMIDINodeName;
00065     uint32_t mGlobalMIDIEventMask;
00066
00067     //Input MIDI Nodes
00068     bool mNeedsInputMIDI;
00069     const char* mInputMIDINodeName;
00070     uint32_t mInputMIDIChannelMask;
00071     int32_t mNumAdditionalInputMIDINodes;
00072
00073     //Transport
00074     bool mNeedsTransport;
00075     const char* mTransportMIDINodeName;
00076
00077     //Meters
00078     int32_t mNumMeters;
00079     const AAX_CTypeID* mMeterIDs;
00080
00081     //Aux Output Stems Feature.
00082     int32_t mNumAuxOutputStems;
00083     const char* mAuxOutputStemNames[kMaxAuxOutputStems];
00084     AAX_EStemFormat mAuxOutputStemFormats[kMaxAuxOutputStems];
00085
00086     //AAX Hybrid
00087     AAX_EStemFormat mHybridInputStemFormat;
00088     AAX_EStemFormat mHybridOutputStemFormat;
00089
00090
00091     //General Properties

```

```

00092     AAX_EStemFormat    mInputStemFormat;
00093     AAX_EStemFormat    mOutputStemFormat;
00094     bool               mUseHostGeneratedGUI;
00095     bool               mCanBypass;
00096     AAX_CTypeID        mManufacturerID;
00097     AAX_CTypeID        mProductID;
00098     AAX_CTypeID        mPluginID;
00099     AAX_CTypeID        mAudiosuiteID;
00100
00107     AAX_CBoolean       mMultiMonoSupport;
00108
00109
00115     AAX_SInstrumentSetupInfo()
00116     {
00117         mNeedsGlobalMIDI = false;
00118         mGlobalMIDINodeName = "GlobalMIDI";
00119         mGlobalMIDIEventMask = 0xffff;
00120         mNeedsInputMIDI = false;
00121         mInputMIDINodeName = "InputMIDI";
00122         mInputMIDIChannelMask = 0xffff;
00123         mNumAdditionalInputMIDINodes = 0;
00124         mNeedsTransport = false;
00125         mTransportMIDINodeName = "Transport";
00126         mNumMeters = 0;
00127         mMeterIDs = 0;
00128         mInputStemFormat = AAX_eStemFormat_Mono;
00129         mOutputStemFormat = AAX_eStemFormat_Mono;
00130         mUseHostGeneratedGUI = false;
00131         mCanBypass = true;
00132         mManufacturerID = 'none';
00133         mProductID = 'none';
00134         mPluginID = 'none';
00135         mAudiosuiteID = 'none';
00136         mMultiMonoSupport = true;
00137
00138         mNumAuxOutputStems = 0;
00139         for (int32_t i=0; i<kMaxAuxOutputStems; i++)
00140         {
00141             mAuxOutputStemNames[i] = 0;
00142             mAuxOutputStemFormats[i] = AAX_eStemFormat_Mono;
00143         }
00144
00145         mHybridInputStemFormat = AAX_eStemFormat_None;
00146         mHybridOutputStemFormat = AAX_eStemFormat_None;
00147     }
00148 };
00149
00150 class AAX_CMonolithicParameters; //predefined for AAX_SInstrumentPrivateData
00156 struct AAX_SInstrumentPrivateData
00157 {
00164     AAX_CMonolithicParameters* mMonolithicParametersPtr;
00165
00166     //<DMT> Removed mOutputStemFormat. Adding it to this structure was not intentional. Please call
    Controller()->GetOutputStemFormat() from your RenderAudio call if you need this info.
00167     //<DMT> Please note, nothing else should be added to this struct. All host information is
    accessible through the AAX_IController in the RenderAudio call.
00168 };
00169
00170
00174 struct AAX_SInstrumentRenderInfo
00175 {
00176     float**                mAudioInputs;
00177     float**                mAudioOutputs;
00178     int32_t*               mNumSamples;
00179     AAX_CTimestamp*        mClock;
00180
00181     AAX_IMIDINode*         mInputNode;
00182     AAX_IMIDINode*         mGlobalNode;
00183     AAX_IMIDINode*         mTransportNode;
00184     AAX_IMIDINode*         mAdditionalInputMIDINodes[kMaxAdditionalMIDINodes];
00185
00186     AAX_SInstrumentPrivateData* mPrivateData;
00187
00188     float**                mMeters;
00189
00190     int64_t*               mCurrentStateNum;
00191 };
00192
00217 class AAX_CMonolithicParameters : public AAX_CEffectParameters
00218 {
00219 public:
00220     AAX_CMonolithicParameters (void);
00221     ~AAX_CMonolithicParameters (void) AAX_OVERRIDE;
00222
00223 protected:
00224     typedef std::pair<AAX_CParamID const, const AAX_IParameterValue*> TParamValPair;
00225

```

```

00241     virtual void      RenderAudio(AAX_SInstrumentRenderInfo* ioRenderInfo, const TParamValPair*
inSynchronizedParamValues[], int32_t inNumSynchronizedParamValues) {}
00243
00258     void              AddSynchronizedParameter(const AAX_IParameter& inParameter);
00260
00261 public:
00268     // Overrides from \ref AAX_CEffectParameters
00269     AAX_Result         UpdateParameterNormalizedValue(AAX_CParamID iParamID, double aValue,
AAX_EUpdateSource inSource) AAX_OVERRIDE;
00270     AAX_Result         GenerateCoefficients() AAX_OVERRIDE;
00271     AAX_Result         ResetFieldData (AAX_CFieldIndex iFieldIndex, void * oData, uint32_t iDataSize) const
AAX_OVERRIDE;
00272     AAX_Result         TimerWakeup() AAX_OVERRIDE;
00281     static AAX_Result  StaticDescribe (AAX_IEffectDescriptor * ioDescriptor, const
AAX_SInstrumentSetupInfo & setupInfo);
00290     static void        AAX_CALLBACK    StaticRenderAudio(AAX_SInstrumentRenderInfo* const
inInstancesBegin [], const void* inInstancesEnd);
00292
00293 private:
00294     // This structure is used on the render thread to set up the contiguous array of TParamValPair*
values
00295     // which is passed to RenderAudio(). The values are drawn from lists of parameter value updates
which
00296     // were queued during GenerateCoefficients()
00297     struct SParamValList
00298     {
00299         // Using 4x the preset queue size: the buffer must be large enough to accommodate the maximum
00300         // number of updates that we expect to be queued between/before executions of the render
callback.
00301         // The maximum queuing that will likely ever occur is during a preset change (i.e. a call to
00302         // SetChunk()), in which updates to all parameters may be queued in the same state frame. It
is
00303         // possible that the host would call SetChunk() on the plug-in more than once before the
render
00304         // callback executes, but probably not more than 2-3x. Therefore 4x seems like a safe upper
limit
00305         // for the capacity of this buffer.
00306         static const int32_t sCap = 4*kSynchronizedParameterQueueSize;
00307
00308         TParamValPair* mElem[sCap];
00309         int32_t mSize;
00310
00311         SParamValList()
00312         {
00313             Clear();
00314         }
00315
00316         void Add(TParamValPair* inElem)
00317         {
00318             AAX_ASSERT(sCap > mSize);
00319             if (sCap > mSize)
00320             {
00321                 mElem[mSize++] = inElem;
00322             }
00323         }
00324
00325         void Append(const SParamValList& inOther)
00326         {
00327             AAX_ASSERT(sCap >= mSize + inOther.mSize);
00328             for (int32_t i = 0; i < inOther.mSize; ++i)
00329             {
00330                 Add(inOther.mElem[i]);
00331             }
00332         }
00333
00334         void Append(const std::list<TParamValPair*>& inOther)
00335         {
00336             AAX_ASSERT(sCap >= mSize + (int64_t)inOther.size());
00337             for (std::list<TParamValPair*>::const_iterator iter = inOther.begin(); iter !=
inOther.end(); ++iter)
00338             {
00339                 Add(*iter);
00340             }
00341         }
00342
00343         void Merge(AAX_IPointerQueue<TParamValPair*>& inOther)
00344         {
00345             do
00346             {
00347                 TParamValPair* const val = inOther.Pop();
00348                 if (NULL == val) { break; }
00349                 Add(val);
00350             } while (1);
00351         }
00352
00353         void Clear()
00354         {

```



```

00355         std::memset(mElem, 0x0, sizeof(mElem));
00356         mSize = 0;
00357     }
00358 };
00359
00360     typedef std::set<const AAX_IParameter*> TParamSet;
00361     typedef std::pair<int64_t, std::list<TParamValPair*> > TNumberedParamStateList;
00362     typedef AAX_CAtomicQueue<TNumberedParamStateList, 256> TNumberedStateListQueue;
00363     typedef AAX_CAtomicQueue<const TParamValPair, 16*kSynchronizedParameterQueueSize>
TParamValPairQueue;
00364
00365
00366     SParamValList GetUpdatesForState(int64_t inTargetStateNum);
00367     void DeleteUsedParameterChanges();
00368
00369 private:
00370     std::set<std::string> mSynchronizedParameters;
00371     int64_t mStateCounter;
00372     TParamSet mDirtyParameters;
00373     TNumberedStateListQueue mQueuedParameterChanges;
00374     TNumberedStateListQueue mFinishedParameterChanges; // Parameter changes ready for
deletion
00375     TParamValPairQueue mFinishedParameterValues; // Parameter values ready for
deletion
00376 };
00377
00378
00379
00380
00381
00382 #endif // AAX_CMONOLITHICPARAMETERS_H

```

15.53 AAX.h File Reference

```

#include <stdint.h>
#include <stddef.h>
#include "AAX_Version.h"
#include "AAX_Enums.h"
#include "AAX_Errors.h"
#include "AAX_Properties.h"
#include <AAX_ALIGN_FILE_BEGIN>
#include <AAX_ALIGN_FILE_HOST>
#include <AAX_ALIGN_FILE_END>
#include <AAX_ALIGN_FILE_RESET>

```

15.53.1 Description

Various utility definitions for AAX.

Classes

- struct [AAX_SPlugInChunkHeader](#)
Plug-in chunk header.
- struct [AAX_SPlugInChunk](#)
Plug-in chunk header + data.
- struct [AAX_SPlugInIdentifierTriad](#)
Plug-in Identifier Triad.
- struct [AAX_CMidiPacket](#)
Packet structure for MIDI data.
- struct [AAX_CMidiStream](#)
MIDI stream data structure used by [AAX_IMIDINode](#).

Macros

C++ compiler macros

- #define `TI_VERSION` 0
Preprocessor flag indicating compilation for TI.
- #define `AAX_CPP11_SUPPORT` 1
Preprocessor toggle for code which requires C++11 compiler support.

C++ keyword macros

Use these macros for keywords which may not be supported on all compilers

Warning

Be careful when using these macros; they are a workaround and the fallback versions of the macros are not guaranteed to provide identical behavior to the fully-supported versions. Always consider the code which will be generated in each case!

If your code is protected with PACE Fusion and you are using a PACE SDK prior to v4 then you must explicitly define `AAX_CPP11_SUPPORT` 0 in your project's preprocessor settings to avoid encountering source failover caused by AAX header includes with exotic syntax.

- #define `AAX_OVERRIDE`
override keyword macro
- #define `AAX_FINAL`
final keyword macro
- #define `AAX_DEFAULT_DTOR(X) ~X() {}`
- #define `AAX_DEFAULT_DTOR_OVERRIDE(X) ~X() {}`
- #define `AAX_DEFAULT_CTOR(X) X() {}`
default keyword macro for a class default constructor
- #define `AAX_DEFAULT_COPY_CTOR(X)`
default keyword macro for a class copy constructor
- #define `AAX_DEFAULT_MOVE_CTOR(X)`
default keyword macro for a class move constructor
- #define `AAX_DEFAULT_ASGN_OPER(X)`
default keyword macro for a class assignment operator
- #define `AAX_DEFAULT_MOVE_OPER(X)`
default keyword macro for a class move-assignment operator
- #define `AAX_DELETE(X) private: X; public:`
delete keyword macro
- #define `AAX_CONSTEXPR` const
constexpr keyword macro
- #define `AAX_UNIQUE_PTR(X) std::auto_ptr<X>`

Pointer definitions

- #define `AAXPointer_32bit` 1
When `AAX_PointerSize == AAXPointer_32bit` this is a 32-bit build.
- #define `AAXPointer_64bit` 2
When `AAX_PointerSize == AAXPointer_64bit` this is a 64-bit build.
- #define `AAX_PointerSize` `AAXPointer_32bit`
Use this definition to check the pointer size in the current build.

Alignment macros

Use these macros to define struct packing alignment for data structures that will be sent across binary or platform boundaries.

```
#include AAX_ALIGN_FILE_BEGIN
#include AAX_ALIGN_FILE_HOST
#include AAX_ALIGN_FILE_END
// Structure definition
#include AAX_ALIGN_FILE_BEGIN
#include AAX_ALIGN_FILE_RESET
#include AAX_ALIGN_FILE_END
```

See the documentation for each macro for individual usage notes and warnings

- `#define AAX_ALIGN_FILE_HOST "AAX_Push2ByteStructAlignment.h"`
Macro to set alignment for data structures that are shared with the host.
- `#define AAX_ALIGN_FILE_ALG "AAX_Push8ByteStructAlignment.h"`
Macro to set alignment for data structures that are used in the alg.
- `#define AAX_ALIGN_FILE_RESET "AAX_PopStructAlignment.h"`
Macro to reset alignment back to default.
- `#define AAX_ALIGN_FILE_BEGIN "AAX_PreStructAlignmentHelper.h"`
Wrapper macro used for warning suppression.
- `#define AAX_ALIGN_FILE_END "AAX_PostStructAlignmentHelper.h"`
- `#define AAX_CALLBACK`
- `#define AAX_PREPROCESSOR_CONCAT_HELPER(X, Y) X ## Y`
- `#define AAX_PREPROCESSOR_CONCAT(X, Y) AAX_PREPROCESSOR_CONCAT_HELPER(X,Y)`
- `#define AAX_FIELD_INDEX(aContextType, aMember) ((AAX_CFieldIndex) (offsetof (aContextType, aMember) / sizeof (void *)))`
Compute the index used to address a context field.
- `typedef int32_t AAX_CIndex`
- `typedef AAX_CIndex AAX_CCount`
- `typedef uint8_t AAX_CBoolean`
Cross-compiler boolean type used by AAX interfaces.
- `typedef uint32_t AAX_CSelector`
- `typedef int64_t AAX_CTimestamp`
Time stamp value. Measured against the DAE clock (see [AAX_IComponentDescriptor::AddClock\(\)](#))
- `typedef int64_t AAX_CTimeOfDay`
Hardware running clock value. MIDI packet time stamps are measured against this clock. This is actually the same as [TransportCounter](#), but kept for compatibility.
- `typedef int64_t AAX_CTransportCounter`
Offset of samples from transport start. Same as [TimeOfDay](#), but added for new interfaces as [TimeOfDay](#) is a confusing name.
- `typedef float AAX_CSampleRate`
Literal sample rate value used by the [sample rate field](#). For [AAX_eProperty_SampleRate](#), use a mask of [AAX_ESampleRateMask](#).
- `typedef uint32_t AAX_CTypeID`
Matches type of [OSType](#) used in classic plugins.
- `typedef int32_t AAX_Result`
- `typedef int32_t AAX_CPropertyValue`
32-bit property values
- `typedef int64_t AAX_CPropertyValue64`
64-bit property values
- `typedef AAX_CPropertyValue AAX_CPointerPropertyValue`
Pointer-sized property values.
- `typedef int32_t AAX_CTargetPlatform`

- Matches type of *target platform*.
- typedef [AAX_CIndex](#) [AAX_CFieldIndex](#)
 - Not used by AAX plug-ins (except in [AAX_FIELD_INDEX](#) macro)
- typedef [AAX_CSelector](#) [AAX_CComponentID](#)
- typedef [AAX_CSelector](#) [AAX_CMeterID](#)
- typedef const char * [AAX_CParamID](#)
 - Parameter identifier.
- typedef [AAX_CParamID](#) [AAX_CPageTableParamID](#)
 - Parameter identifier used in a page table.
- typedef const char * [AAX_CEffectID](#)
 - URL-style Effect identifier. Must be unique among all registered effects in the collection.
- typedef [_acfUID](#) [acfUID](#)
- typedef [acfUID](#) [AAX_Feature_UID](#)
- typedef const float *const * [AAX_CAudioInPort](#)
 - AAX algorithm audio input port data type
- typedef float *const * [AAX_CAudioOutPort](#)
 - AAX algorithm audio output port data type
- typedef float *const [AAX_CMeterPort](#)
 - AAX algorithm meter port data type
- typedef struct [AAX_SPlugInChunkHeader](#) [AAX_SPlugInChunkHeader](#)
- typedef struct [AAX_SPlugInChunk](#) [AAX_SPlugInChunk](#)
- typedef struct [AAX_SPlugInChunk](#) * [AAX_SPlugInChunkPtr](#)
- typedef struct [AAX_SPlugInIdentifierTriad](#) [AAX_SPlugInIdentifierTriad](#)
- typedef struct [AAX_SPlugInIdentifierTriad](#) * [AAX_SPlugInIdentifierTriadPtr](#)
- [AAX_CONSTEXPR](#) size_t [kAAX_ParameterIdentifierMaxSize](#) = 32
- [AAX_CBoolean](#) [sampleRateInMask](#) ([AAX_CSampleRate](#) inSR, uint32_t iMask)
 - Determines whether a particular [AAX_CSampleRate](#) is present in a given mask of [AAX_ESampleRateMask](#).
- [AAX_CSampleRate](#) [getLowestSampleRateInMask](#) (uint32_t iMask)
 - Converts from a mask of [AAX_ESampleRateMask](#) to the lowest supported [AAX_CSampleRate](#) value in Hz.
- uint32_t [getMaskForSampleRate](#) (float inSR)
 - Returns the [AAX_ESampleRateMask](#) selector for a literal sample rate.

15.53.2 Macro Definition Documentation

15.53.2.1 TI_VERSION

```
#define TI_VERSION 0
```

Preprocessor flag indicating compilation for TI.

15.53.2.2 AAX_CPP11_SUPPORT

```
#define AAX_CPP11_SUPPORT 1
```

Preprocessor toggle for code which requires C++11 compiler support.

15.53.2.3 AAX_OVERRIDE

```
#define AAX_OVERRIDE
```

override keyword macro

15.53.2.4 AAX_FINAL

```
#define AAX_FINAL
```

final keyword macro

15.53.2.5 AAX_DEFAULT_DTOR

```
#define AAX_DEFAULT_DTOR(  
    X ) ~X() {}
```

15.53.2.6 AAX_DEFAULT_DTOR_OVERRIDE

```
#define AAX_DEFAULT_DTOR_OVERRIDE(  
    X ) ~X() {}
```

15.53.2.7 AAX_DEFAULT_CTOR

```
#define AAX_DEFAULT_CTOR(  
    X ) X() {}
```

default keyword macro for a class default constructor

15.53.2.8 AAX_DEFAULT_COPY_CTOR

```
#define AAX_DEFAULT_COPY_CTOR(  
    X )
```

default keyword macro for a class copy constructor

15.53.2.9 AAX_DEFAULT_MOVE_CTOR

```
#define AAX_DEFAULT_MOVE_CTOR(  
    X )
```

default keyword macro for a class move constructor

15.53.2.10 AAX_DEFAULT_ASGN_OPER

```
#define AAX_DEFAULT_ASGN_OPER(  
    X )
```

default keyword macro for a class assignment operator

15.53.2.11 AAX_DEFAULT_MOVE_OPER

```
#define AAX_DEFAULT_MOVE_OPER(  
    X )
```

default keyword macro for a class move-assignment operator

15.53.2.12 AAX_DELETE

```
#define AAX_DELETE(  
    X ) private: X; public:
```

delete keyword macro

Warning

The non-C++11 version of this macro assumes `public` declaration access

15.53.2.13 AAX_CONSTEXPR

```
#define AAX_CONSTEXPR const
```

constexpr keyword macro

15.53.2.14 AAX_UNIQUE_PTR

```
#define AAX_UNIQUE_PTR(  
    X ) std::auto_ptr<X>
```

15.53.2.15 AAXPointer_32bit

```
#define AAXPointer_32bit 1
```

When `AAX_PointerSize == AAXPointer_32bit` this is a 32-bit build.

15.53.2.16 AAXPointer_64bit

```
#define AAXPointer_64bit 2
```

When `AAX_PointerSize == AAXPointer_64bit` this is a 64-bit build.

15.53.2.17 AAX_PointerSize

```
#define AAX_PointerSize AAXPointer\_32bit
```

Use this definition to check the pointer size in the current build.

See also

[AAXPointer_32bit](#)

[AAXPointer_64bit](#)

15.53.2.18 AAX_ALIGN_FILE_HOST

```
#define AAX_ALIGN_FILE_HOST "AAX_Push2ByteStructAlignment.h"
```

Macro to set alignment for data structures that are shared with the host.

This macro is used to set alignment for data structures that are part of the AAX ABI. You should not need to use this macro for any custom data structures in your plug-in.

15.53.2.19 AAX_ALIGN_FILE_ALG

```
#define AAX_ALIGN_FILE_ALG "AAX_Push8ByteStructAlignment.h"
```

Macro to set alignment for data structures that are used in the alg.

IMPORTANT: Be very careful to maintain correct data alignment when sending data structures between platforms.

Warning

- This macro does not guarantee data alignment compatibility for data structures which include base classes/structs or virtual functions. The MSVC, GCC and LLVM/clang, and CCS (TI) compilers do not support data structure cross-compatibility for these types of structures. clang will now present a warning when these macros are used on any such structures: `#pragma ms_struct` can not be used with dynamic classes or structures
- Struct Member Alignment (/Zp) on Microsoft compilers must be set to a minimum of 8-byte packing in order for this macro to function properly. For more information, see this MSDN article:
<http://msdn.microsoft.com/en-us/library/ms253935.aspx>

15.53.2.20 AAX_ALIGN_FILE_RESET

```
#define AAX_ALIGN_FILE_RESET "AAX_PopStructAlignment.h"
```

Macro to reset alignment back to default.

15.53.2.21 AAX_ALIGN_FILE_BEGIN

```
#define AAX_ALIGN_FILE_BEGIN "AAX_PreStructAlignmentHelper.h"
```

Wrapper macro used for warning suppression.

This wrapper is required in llvm 10.0 and later due to the addition of the `-Wpragma-pack` warning. This is a useful compiler warning but it is awkward to properly suppress in cases where we are intentionally including only part of the push/pop sequence in a single file, as with the `AAX_ALIGN_FILE_XXX` macros.

15.53.2.22 AAX_ALIGN_FILE_END

```
#define AAX_ALIGN_FILE_END "AAX_PostStructAlignmentHelper.h"
```

Wrapper macro used for warning suppression.

This wrapper is required in llvm 10.0 and later due to the addition of the `-Wpragma-pack` warning. This is a useful compiler warning but it is awkward to properly suppress in cases where we are intentionally including only part of the push/pop sequence in a single file, as with the `AAX_ALIGN_FILE_XXX` macros.

15.53.2.23 AAX_CALLBACK

```
#define AAX_CALLBACK
```

15.53.2.24 AAX_PREPROCESSOR_CONCAT_HELPER

```
#define AAX_PREPROCESSOR_CONCAT_HELPER(  
    X,  
    Y ) X ## Y
```

15.53.2.25 AAX_PREPROCESSOR_CONCAT

```
#define AAX_PREPROCESSOR_CONCAT(  
    X,  
    Y ) AAX_PREPROCESSOR_CONCAT_HELPER(X,Y)
```

15.53.2.26 AAX_FIELD_INDEX

```
#define AAX_FIELD_INDEX(  
    aContextType,  
    aMember ) ((AAX_CFieldIndex) (offsetof (aContextType, aMember) / sizeof (void  
*)) )
```

Compute the index used to address a context field.

This macro expands to a constant expression suitable for use in enumerator definitions and case labels so `int32_t` as `aMember` is a constant specifier.

Parameters

in	<i>aContextType</i>	The name of context type
in	<i>aMember</i>	The name or other specifier of a field of that context type

15.53.3 Typedef Documentation

15.53.3.1 AAX_CIndex

```
typedef int32_t AAX_CIndex
```

Todo Not used by AAX plug-ins (except as [AAX_CFieldIndex](#))

15.53.3.2 AAX_CCount

```
typedef AAX\_CIndex AAX\_CCount
```

Todo Not used by AAX plug-ins

15.53.3.3 AAX_CBoolean

```
typedef uint8_t AAX\_CBoolean
```

Cross-compiler boolean type used by AAX interfaces.

15.53.3.4 AAX_CSelector

```
typedef uint32_t AAX\_CSelector
```

Todo Clean up usage; currently used for a variety of ID-related values

15.53.3.5 AAX_CTimestamp

```
typedef int64_t AAX\_CTimestamp
```

Time stamp value. Measured against the DAE clock (see [AAX_IComponentDescriptor::AddClock\(\)](#))

15.53.3.6 AAX_CTimeOfDay

```
typedef int64_t AAX\_CTimeOfDay
```

Hardware running clock value. MIDI packet time stamps are measured against this clock. This is actually the same as TransportCounter, but kept for compatibility.

15.53.3.7 AAX_CTransportCounter

```
typedef int64_t AAX_CTransportCounter
```

Offset of samples from transport start. Same as TimeOfDay, but added for new interfaces as TimeOfDay is a confusing name.

15.53.3.8 AAX_CSampleRate

```
typedef float AAX_CSampleRate
```

Literal sample rate value used by the [sample rate field](#). For [AAX_eProperty_SampleRate](#), use a mask of [AAX_ESampleRateMask](#).

See also

[sampleRateInMask](#)

15.53.3.9 AAX_CTypeID

```
typedef uint32_t AAX_CTypeID
```

Matches type of OSType used in classic plugins.

15.53.3.10 AAX_Result

```
typedef int32_t AAX_Result
```

15.53.3.11 AAX_CPropertyValue

```
typedef int32_t AAX_CPropertyValue
```

32-bit property values

Use this property value type for all properties unless otherwise specified by the property documentation

15.53.3.12 AAX_CPropertyValue64

```
typedef int64_t AAX_CPropertyValue64
```

64-bit property values

Do not use this value type unless specified explicitly in the property documentation

15.53.3.13 AAX_CPointerPropertyValue

```
typedef AAX_CPropertyValue AAX_CPointerPropertyValue
```

Pointer-sized property values.

Do not use this value type unless specified explicitly in the property documentation

15.53.3.14 AAX_CTargetPlatform

```
typedef int32_t AAX_CTargetPlatform
```

Matches type of [target platform](#).

15.53.3.15 AAX_CFieldIndex

```
typedef AAX_CIndex AAX_CFieldIndex
```

Not used by AAX plug-ins (except in [AAX_FIELD_INDEX](#) macro)

15.53.3.16 AAX_CComponentID

```
typedef AAX_CSelector AAX_CComponentID
```

Todo Not used by AAX plug-ins

15.53.3.17 AAX_CMeterID

```
typedef AAX_CSelector AAX_CMeterID
```

Todo Not used by AAX plug-ins

15.53.3.18 AAX_CParamID

```
typedef const char* AAX_CParamID
```

Parameter identifier.

Note

While this is a string, it must be less than 32 characters in length. (strlen of 31 or less)

See also

[kAAX_ParameterIdentifierMaxSize](#)

15.53.3.19 AAX_CPageTableParamID

```
typedef AAX_CParamID AAX_CPageTableParamID
```

Parameter identifier used in a page table.

May be a parameter ID or a parameter name string depending on the page table formatting. Must be less than 32 characters in length (strlen of 31 or less.)

See also

[Parameter identifiers](#) in the [Page Table Guide](#)

15.53.3.20 AAX_CEffectID

```
typedef const char* AAX_CEffectID
```

URL-style Effect identifier. Must be unique among all registered effects in the collection.

15.53.3.21 acfUID

```
typedef _acfUID acfUID
```

15.53.3.22 AAX_Feature_UID

```
typedef acfUID AAX_Feature_UID
```

Identifier for AAX features

See [AAX_IDescriptionHost::AcquireFeatureProperties\(\)](#) and [AAX_IFeatureInfo](#)

15.53.3.23 AAX_CAudioInPort

```
typedef const float* const* AAX_CAudioInPort
```

AAX algorithm audio input port data type

Audio input ports are provided with a pointer to an array of const audio buffers, with one buffer provided per input or side chain channel.

Todo Not used directly by AAX plug-ins

15.53.3.24 AAX_CAudioOutPort

```
typedef float* const* AAX_CAudioOutPort
```

AAX algorithm audio output port data type

Audio output ports are provided with a pointer to an array of audio buffers, with one buffer provided per output or auxiliary output channel.

Todo Not used directly by AAX plug-ins

15.53.3.25 AAX_CMeterPort

```
typedef float* const AAX_CMeterPort
```

AAX algorithm meter port data type

Meter output ports are provided with a pointer to an array of floats, with one float provided per meter tap. The algorithm is responsible for setting these to the corresponding per-buffer peak sample values.

Todo Not used directly by AAX plug-ins

15.53.3.26 AAX_SPlugInChunkHeader

```
typedef struct AAX_SPlugInChunkHeader AAX_SPlugInChunkHeader
```

15.53.3.27 AAX_SPlugInChunk

```
typedef struct AAX_SPlugInChunk AAX_SPlugInChunk
```

15.53.3.28 AAX_SPlugInChunkPtr

```
typedef struct AAX_SPlugInChunk * AAX_SPlugInChunkPtr
```

15.53.3.29 AAX_SPlugInIdentifierTriad

```
typedef struct AAX_SPlugInIdentifierTriad AAX_SPlugInIdentifierTriad
```

15.53.3.30 AAX_SPlugInIdentifierTriadPtr

```
typedef struct AAX_SPlugInIdentifierTriad * AAX_SPlugInIdentifierTriadPtr
```

15.53.4 Function Documentation

15.53.4.1 sampleRateInMask()

```
AAX_CBoolean sampleRateInMask (  
    AAX_CSampleRate inSR,  
    uint32_t iMask ) [inline]
```

Determines whether a particular [AAX_CSampleRate](#) is present in a given mask of [AAX_ESampleRateMask](#).

See also

[kAAX_Property_SampleRate](#)

References [AAX_eSampleRateMask_176400](#), [AAX_eSampleRateMask_192000](#), [AAX_eSampleRateMask_44100](#), [AAX_eSampleRateMask_48000](#), [AAX_eSampleRateMask_88200](#), and [AAX_eSampleRateMask_96000](#).

15.53.4.2 `getLowestSampleRateInMask()`

```
AAX_CSampleRate getLowestSampleRateInMask (
    uint32_t iMask ) [inline]
```

Converts from a mask of [AAX_ESampleRateMask](#) to the lowest supported [AAX_CSampleRate](#) value in Hz.

References [AAX_eSampleRateMask_176400](#), [AAX_eSampleRateMask_192000](#), [AAX_eSampleRateMask_44100](#), [AAX_eSampleRateMask_48000](#), [AAX_eSampleRateMask_88200](#), and [AAX_eSampleRateMask_96000](#).

15.53.4.3 `getMaskForSampleRate()`

```
uint32_t getMaskForSampleRate (
    float inSR ) [inline]
```

Returns the [AAX_ESampleRateMask](#) selector for a literal sample rate.

The given rate must be an exact match with one of the available selectors. If no exact match is found then [AAX_eSampleRateMask_No](#) is returned.

References [AAX_eSampleRateMask_176400](#), [AAX_eSampleRateMask_192000](#), [AAX_eSampleRateMask_44100](#), [AAX_eSampleRateMask_48000](#), [AAX_eSampleRateMask_88200](#), [AAX_eSampleRateMask_96000](#), and [AAX_eSampleRateMask_No](#).

15.53.5 Variable Documentation

15.53.5.1 `kAAX_ParameterIdentifierMaxSize`

```
AAX_CONSTEXPR size_t kAAX_ParameterIdentifierMaxSize = 32
```

Maximum size for a [AAX_CParamID](#) including the null-terminating character

15.54 AAX.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003
00004   * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012
00025  /*=====*/
00026
00027
00028  #pragma once
00029
00031  #ifndef _AAX_H_
00032  #define _AAX_H_
00033
00034  #include <stdint.h>
00035  #include <stddef.h>
00036
00037  #include "AAX_Version.h"
00038  #include "AAX_Enums.h"
00039  #include "AAX_Errors.h"
00040  #include "AAX_Properties.h"
00041
00042
00043
00044
00055
00056  #ifndef TI_VERSION
00057      #if defined _TMS320C6X
00058          #define TI_VERSION 1
00059      #elif defined DOXYGEN_PREPROCESSOR
00060          #define TI_VERSION 0
00061      #endif
00062  #endif
00063
00064
00065  #ifndef AAX_CPP11_SUPPORT
00066      #if (defined __cplusplus) && (__cplusplus >= 201103L)
00067          #define AAX_CPP11_SUPPORT 1
00068          // VS2015 supports all features except expression SFINAE
00069      #elif ((defined _MSVC_LANG) && (_MSVC_LANG >= 201402))
00070          #define AAX_CPP11_SUPPORT 1
00071          // Let Doxygen see the C++11 version of all code
00072      #elif defined DOXYGEN_PREPROCESSOR
00073          #define AAX_CPP11_SUPPORT 1
00074      #endif
00075  #endif
00076
00077
00136
00137  #if AAX_CPP11_SUPPORT
00138      # define AAX_OVERRIDE override
00139      # define AAX_FINAL final
00140      # define AAX_DEFAULT_DTOR(X) ~X() = default
00141      # define AAX_DEFAULT_DTOR_OVERRIDE(X) ~X() override = default
00142      # define AAX_DEFAULT_CTOR(X) X() = default
00143      # define AAX_DEFAULT_COPY_CTOR(X) X(const X&) = default
00144      # define AAX_DEFAULT_ASGN_OPER(X) X& operator=(const X&) = default
00145      # define AAX_DELETE(X) X = delete
00146      # define AAX_DEFAULT_MOVE_CTOR(X) X(X&) = default
00147      # define AAX_DEFAULT_MOVE_OPER(X) X& operator=(X&) = default
00148      # define AAX_CONSTEXPR constexpr
00149      # define AAX_UNIQUE_PTR(X) std::unique_ptr<X>
00150  #else
00151      # define AAX_OVERRIDE
00152      # define AAX_FINAL
00153      # define AAX_DEFAULT_DTOR(X) ~X() {}
00154      # define AAX_DEFAULT_DTOR_OVERRIDE(X) ~X() {}
00155      # define AAX_DEFAULT_CTOR(X) X() {}
00156      # define AAX_DEFAULT_COPY_CTOR(X)
00157      # define AAX_DEFAULT_MOVE_CTOR(X)
00158      # define AAX_DEFAULT_ASGN_OPER(X)
00159      # define AAX_DEFAULT_MOVE_OPER(X)
00160      // Assumes public access in the declaration scope where AAX_DELETE is used
00161      # define AAX_DELETE(X) private: X; public:
00162      # define AAX_CONSTEXPR const
00163      # define AAX_UNIQUE_PTR(X) std::auto_ptr<X>
00164  #endif

```

```

00165
00166
00183 #define AAXPointer_32bit    1
00184 #define AAXPointer_64bit    2
00185
00186 #if !defined(AAX_PointerSize)
00187     #if defined(_M_X64) || defined (__LP64__)
00188         #define AAX_PointerSize AAXPointer_64bit
00189     #else
00190         #define AAX_PointerSize AAXPointer_32bit
00191     #endif
00192 #endif
00193
00194 // ensure that preprocessor comparison logic gives the correct result
00195 #if ((AAX_PointerSize == AAXPointer_32bit) && (defined(_M_X64) || defined (__LP64__)))
00196     #error incorrect result of AAX_PointerSize check!
00197 #elif ((AAX_PointerSize == AAXPointer_64bit) && !(defined(_M_X64) || defined (__LP64__)))
00198     #error incorrect result of AAX_PointerSize check!
00199 #endif
00200
00201
00263
00264 #if ( defined(_WIN64) || defined(__LP64__) )
00265     #define AAX_ALIGN_FILE_HOST    "AAX_Push8ByteStructAlignment.h"
00266 #elif ( defined(_TMS320C6X) )
00267     // AAX_ALIGN_FILE_HOST is not compatible with this compiler
00268     // We don't use an #error here b/c that causes Doxygen to get confused
00269 #else
00270     #define AAX_ALIGN_FILE_HOST    "AAX_Push2ByteStructAlignment.h"
00271 #endif
00272 #define AAX_ALIGN_FILE_ALG    "AAX_Push8ByteStructAlignment.h"
00273 #define AAX_ALIGN_FILE_RESET    "AAX_PopStructAlignment.h"
00274 #define AAX_ALIGN_FILE_BEGIN    "AAX_PreStructAlignmentHelper.h"
00275 #define AAX_ALIGN_FILE_END    "AAX_PostStructAlignmentHelper.h"
00276
00277
00278 #ifndef AAX_CALLBACK
00279 #   ifdef _MSC_VER
00280 #       define AAX_CALLBACK    __cdecl
00281 #   else
00282 #       define AAX_CALLBACK
00283 #   endif
00284 #endif // AAX_CALLBACK
00285
00286
00287 #ifdef _MSC_VER
00288 #   define AAX_RESTRICT
00289 #elif defined(_TMS320C6X) // TI
00290 #   define AAX_RESTRICT restrict
00291 #elif defined (__GNUC__) // Mac
00292 #   define AAX_RESTRICT __restrict__
00293 #endif // _MSC_VER
00294
00295
00296 // preprocessor helper macros
00297 #define AAX_PREPROCESSOR_CONCAT_HELPER(X,Y) X ## Y
00298 #define AAX_PREPROCESSOR_CONCAT(X,Y) AAX_PREPROCESSOR_CONCAT_HELPER(X,Y)
00299
00300
00301
00302 #ifdef _MSC_VER
00303 // Disable unknown pragma warning for TI pragmas under VC++
00304 #pragma warning( disable : 4068 )
00305 #endif
00306
00307
00320 #define AAX_FIELD_INDEX( aContextType, aMember ) \
00321     ((AAX_CFieldIndex) (offsetof (aContextType, aMember) / sizeof (void *)))
00322
00323
00324 typedef int32_t        AAX_CIndex;
00325 typedef AAX_CIndex     AAX_CCount;
00326 typedef uint8_t        AAX_CBoolean;
00327 typedef uint32_t       AAX_CSelector;
00328 typedef int64_t        AAX_CTimestamp;
00329 typedef int64_t        AAX_CTimeOfDay;
00330 typedef int64_t        AAX_CTransportCounter;
00331 typedef float          AAX_CSampleRate;
00332
00333 typedef uint32_t       AAX_CTypeID;
00334 typedef int32_t        AAX_Result;
00335 typedef int32_t        AAX_CPropertyValue;
00336 typedef int64_t        AAX_CPropertyValue64;
00337 #if AAX_PointerSize == AAXPointer_32bit
00338     typedef AAX_CPropertyValue AAX_CPointerPropertyValue;
00339 #elif AAX_PointerSize == AAXPointer_64bit
00340     typedef AAX_CPropertyValue64 AAX_CPointerPropertyValue;

```

```

00341 #else
00342     #error unexpected pointer size
00343 #endif
00344 typedef int32_t      AAX_CTargetPlatform;
00345
00346 typedef AAX_CIndex    AAX_CFieldIndex;
00347 typedef AAX_CSelector AAX_CComponentID;
00348 typedef AAX_CSelector AAX_CMeterID;
00349 typedef const char *   AAX_CParamID;
00350 typedef AAX_CParamID   AAX_CPageTableParamID;
00351 typedef const char *   AAX_CEffectID;
00352
00353 // Forward declarations required for AAX_Feature_UID typedef (the "real" typedef is in AAX_UIDs.h)
00354 struct _acfUID;
00355 typedef _acfUID acfUID;
00356
00361 typedef acfUID AAX_Feature_UID;
00362
00364 AAX_CONSTEXPR size_t      kAAX_ParameterIdentifierMaxSize = 32;
00365
00366 static const AAX_CTimestamp kAAX_Never = (AAX_CTimestamp) ~0ULL;
00367
00368
00370 #ifdef _TMS320C6X
00371     // TI's C compiler defaults to 8 byte alignment of doubles
00372     #define AAX_ALIGNED(v)
00373 #elif defined(__GNUC__)
00374     #define AAX_ALIGNED(v) __attribute__((aligned(v)))
00375 #elif defined(_MSC_VER)
00376     #define AAX_ALIGNED(v) __declspec(align(v))
00377 #else
00378     #error Teach me to align data types with this compiler.
00379 #endif
00380
00381
00387 static
00388 inline
00389 int32_t
00390 AAX_GetStemFormatChannelCount (
00391     AAX_EStemFormat    inStemFormat)
00392 {
00393     return AAX_STEM_FORMAT_CHANNEL_COUNT (inStemFormat);
00394 }
00395
00396
00406 typedef const float * const *      AAX_CAudioInPort;
00407
00408
00418 typedef float * const *      AAX_CAudioOutPort;
00419
00420
00431 typedef float * const      AAX_CMeterPort;
00432
00433
00440 inline AAX_CBoolean sampleRateInMask(AAX_CSampleRate inSR, uint32_t iMask)
00441 {
00442     return static_cast<AAX_CBoolean>(
00443         (44100.0 == inSR) ? ((iMask & AAX_eSampleRateMask_44100) != 0) :
00444         (48000.0 == inSR) ? ((iMask & AAX_eSampleRateMask_48000) != 0) :
00445         (88200.0 == inSR) ? ((iMask & AAX_eSampleRateMask_88200) != 0) :
00446         (96000.0 == inSR) ? ((iMask & AAX_eSampleRateMask_96000) != 0) :
00447         (176400.0 == inSR) ? ((iMask & AAX_eSampleRateMask_176400) != 0) :
00448         (192000.0 == inSR) ? ((iMask & AAX_eSampleRateMask_192000) != 0) : false
00449     );
00450 }
00451
00456 inline AAX_CSampleRate getLowestSampleRateInMask(uint32_t iMask)
00457 {
00458     return (
00459         ((iMask & AAX_eSampleRateMask_44100) != 0) ? 44100.0f : // AAX_eSampleRateMask_All returns
00460         ((iMask & AAX_eSampleRateMask_48000) != 0) ? 48000.0f :
00461         ((iMask & AAX_eSampleRateMask_88200) != 0) ? 88200.0f :
00462         ((iMask & AAX_eSampleRateMask_96000) != 0) ? 96000.0f :
00463         ((iMask & AAX_eSampleRateMask_176400) != 0) ? 176400.0f :
00464         ((iMask & AAX_eSampleRateMask_192000) != 0) ? 192000.0f : 0.0f
00465     );
00466 }
00467
00475 inline uint32_t getMaskForSampleRate(float inSR)
00476 {
00477     return (
00478         (44100.0 == inSR) ? AAX_eSampleRateMask_44100 :
00479         (48000.0 == inSR) ? AAX_eSampleRateMask_48000 :
00480         (88200.0 == inSR) ? AAX_eSampleRateMask_88200 :
00481         (96000.0 == inSR) ? AAX_eSampleRateMask_96000 :
00482         (176400.0 == inSR) ? AAX_eSampleRateMask_176400 :

```

```

00483         (192000.0 == inSR) ? AAX_eSampleRateMask_192000 : AAX_eSampleRateMask_No
00484     );
00485 }
00486
00487
00488 #ifndef _TMS320C6X
00489
00490 #include AAX_ALIGN_FILE_BEGIN
00491 #include AAX_ALIGN_FILE_HOST
00492 #include AAX_ALIGN_FILE_END
00493
00494 #endif
00495
00521 struct AAX_SPlugInChunkHeader {
00522     int32_t          fSize;
00523     int32_t          fVersion;
00524     AAX_CTypeID      fManufacturerID;
00525     AAX_CTypeID      fProductID;
00526     AAX_CTypeID      fPlugInID;
00527     AAX_CTypeID      fChunkID;
00528     unsigned char    fName[32];
00529 };
00530 typedef struct AAX_SPlugInChunkHeader AAX_SPlugInChunkHeader;
00531
00536 struct AAX_SPlugInChunk {
00537     int32_t          fSize;
00538     int32_t          fVersion;
00539     AAX_CTypeID      fManufacturerID;
00540     AAX_CTypeID      fProductID;
00541     AAX_CTypeID      fPlugInID;
00542     AAX_CTypeID      fChunkID;
00543     unsigned char    fName[32];
00544     char             fData[1];
00545 };
00546 typedef struct AAX_SPlugInChunk AAX_SPlugInChunk, *AAX_SPlugInChunkPtr;
00547
00553 struct AAX_SPlugInIdentifierTriad {
00554     AAX_CTypeID      mManufacturerID;
00555     AAX_CTypeID      mProductID;
00556     AAX_CTypeID      mPlugInID;
00557 };
00558 typedef struct AAX_SPlugInIdentifierTriad AAX_SPlugInIdentifierTriad, *AAX_SPlugInIdentifierTriadPtr;
00559
00560 #ifndef _TMS320C6X
00561 #include AAX_ALIGN_FILE_BEGIN
00562 #include AAX_ALIGN_FILE_RESET
00563 #include AAX_ALIGN_FILE_END
00564 #endif
00565
00566 #ifndef TI_VERSION
00567 static inline bool operator==(const AAX_SPlugInIdentifierTriad& v1, const AAX_SPlugInIdentifierTriad&
00568 v2)
00569 {
00570     return ((v1.mManufacturerID == v2.mManufacturerID) &&
00571             (v1.mProductID == v2.mProductID) &&
00572             (v1.mPlugInID == v2.mPlugInID));
00573 }
00574 static inline bool operator!=(const AAX_SPlugInIdentifierTriad& v1, const AAX_SPlugInIdentifierTriad&
00575 v2)
00576 {
00577     return false == operator==(v1, v2);
00578 }
00579 static inline bool operator<(const AAX_SPlugInIdentifierTriad & lhs, const AAX_SPlugInIdentifierTriad
00580 & rhs)
00581 {
00582     if (lhs.mManufacturerID < rhs.mManufacturerID)
00583         return true;
00584     if (lhs.mManufacturerID == rhs.mManufacturerID)
00585     {
00586         if (lhs.mProductID < rhs.mProductID)
00587             return true;
00588         if (lhs.mProductID == rhs.mProductID)
00589             if (lhs.mPlugInID < rhs.mPlugInID)
00590                 return true;
00591     }
00592     return false;
00593 }
00594 }
00595
00596 static inline bool operator>=(const AAX_SPlugInIdentifierTriad & lhs, const AAX_SPlugInIdentifierTriad
00597 & rhs)
00598 {
00599     return false == operator<(lhs, rhs);
00600 }
00601 }

```

```

00600
00601 static inline bool operator>(const AAX_SPlugInIdentifierTriad & lhs, const AAX_SPlugInIdentifierTriad
& rhs)
00602 {
00603     return operator>=(lhs, rhs) && operator!=(lhs, rhs);
00604 }
00605
00606 static inline bool operator<=(const AAX_SPlugInIdentifierTriad & lhs, const AAX_SPlugInIdentifierTriad
& rhs)
00607 {
00608     return false == operator>(lhs, rhs);
00609 }
00610 #endif //TI_VERSION
00611
00612
00613 //<DMT> For historical compatibility with PT10, we have to make the MIDI structures DEFAULT aligned
instead of ALG aligned. With PT11 and 64 bit, these will now be ALG aligned.
00614 #if ( defined(_WIN64) || defined(__LP64__) || defined(_TMS320C6X) )
00615     #include AAX_ALIGN_FILE_BEGIN
00616     #include AAX_ALIGN_FILE_ALG
00617     #include AAX_ALIGN_FILE_END
00618 #else
00619     #if defined (__GNUC__)
00620         #pragma options align=power // To maintain backwards-compatibility with pre-10 versions of Pro
Tools
00621     #else // Windows, other
00622         #include AAX_ALIGN_FILE_BEGIN
00623         #include AAX_ALIGN_FILE_HOST
00624         #include AAX_ALIGN_FILE_END
00625     #endif
00626 #endif
00627
00635 struct AAX_CMidiPacket
00636 {
00637     uint32_t          mTimestamp;
00638     uint32_t          mLength;
00639     unsigned char     mData[4];
00640     AAX_CBoolean      mIsImmediate;
00641 };
00642
00660 struct AAX_CMidiStream
00661 {
00662     uint32_t          mBufferSize;
00663     AAX_CMidiPacket*  mBuffer;
00664 };
00665
00666 #if ( defined(_WIN64) || defined(__LP64__) || defined(_TMS320C6X) )
00667     #include AAX_ALIGN_FILE_BEGIN
00668     #include AAX_ALIGN_FILE_RESET
00669     #include AAX_ALIGN_FILE_END
00670 #else
00671     #if defined (__GNUC__)
00672         #pragma pack() // To maintain backwards-compatibility with pre-10 versions of Pro Tools
00673     #else
00674         #include AAX_ALIGN_FILE_BEGIN
00675         #include AAX_ALIGN_FILE_RESET
00676         #include AAX_ALIGN_FILE_END
00677     #endif
00678 #endif
00679
00681 #endif // #ifndef _AAX_H_

```

15.55 AAX_Assert.h File Reference

```

#include "AAX_Enums.h"
#include "AAX_CHostServices.h"

```

15.55.1 Description

Declarations for cross-platform AAX_ASSERT, AAX_TRACE and related facilities.

- **AAX_ASSERT(condition)** - If the condition is `false` triggers some manner of warning, e.g. a dialog in a developer build or a DigiTrace log in a shipping build. May be used on host or TI.

- [AAX_DEBUGASSERT\(condition \)](#) - Variant of [AAX_ASSERT](#) which is only active in debug builds of the plug-in.
- [AAX_TRACE_RELEASE\(iPriority, iMessageStr \[,params...\] \)](#) - Traces a printf- style message to the DigiTrace log file. Enabled using the `DTF_AAXPLUGINS` DigiTrace facility.
- [AAX_TRACE\(iPriority, iMessageStr \[, params...\] \)](#) - Variant of [AAX_TRACE_RELEASE](#) which only emits logs in debug builds of the plug-in.
- [AAX_STACKTRACE_RELEASE\(iPriority, iMessageStr \[,params...\] \)](#) - Similar to [AAX_TRACE_RELEASE](#) but prints a stack trace as well as a log message
- [AAX_STACKTRACE\(iPriority, iMessageStr \[,params...\] \)](#) - Variant of [AAX_STACKTRACE_RELEASE](#) which only emits logs in debug builds of the plug-in.
- [AAX_TRACEORSTACKTRACE_RELEASE\(iTracePriority, iStackTracePriority, iMessageStr \[,params...\] \)](#) - Combination of [AAX_TRACE_RELEASE](#) and [AAX_STACKTRACE_RELEASE](#); a stack trace is emitted if logging is enabled at `iStackTracePriority`. Otherwise, if logging is enabled at `iTracePriority` then emits a log.

For all trace macros:

`iPriority` is one of

- [kAAX_Trace_Priority_Low](#)
- [kAAX_Trace_Priority_Normal](#)
- [kAAX_Trace_Priority_High](#)
- [kAAX_Trace_Priority_Critical](#)

These correspond to how the trace messages are filtered using [DigiTrace](#).

Note

Disabling the `DTF_AAXPLUGINS` facility will slightly reduce the overhead of trace statements and chip communication on HDX systems.

=====

Macros

- `#define kAAX_Trace_Priority_None AAX_eTracePriorityHost_None`
- `#define kAAX_Trace_Priority_Critical AAX_eTracePriorityHost_Critical`
- `#define kAAX_Trace_Priority_High AAX_eTracePriorityHost_High`
- `#define kAAX_Trace_Priority_Normal AAX_eTracePriorityHost_Normal`
- `#define kAAX_Trace_Priority_Low AAX_eTracePriorityHost_Low`
- `#define kAAX_Trace_Priority_Lowest AAX_eTracePriorityHost_Lowest`
- `#define AAX_TRACE_RELEASE(iPriority, ...)`
Print a trace statement to the log.
- `#define AAX_STACKTRACE_RELEASE(iPriority, ...)`
Print a stack trace statement to the log.
- `#define AAX_TRACEORSTACKTRACE_RELEASE(iTracePriority, iStackTracePriority, ...)`
Print a trace statement with an optional stack trace to the log.
- `#define AAX_ASSERT(condition)`

- Asserts that a condition is true and logs an error if the condition is false.*
- #define `AAX_DEBUGASSERT`(condition) do { ; } while (0)
 - Asserts that a condition is true and logs an error if the condition is false (debug plug-in builds only)*
- #define `AAX_TRACE`(iPriority, ...) do { ; } while (0)
 - Print a trace statement to the log (debug plug-in builds only)*
- #define `AAX_STACKTRACE`(iPriority, ...) do { ; } while (0)
 - Print a stack trace statement to the log (debug builds only)*
- #define `AAX_TRACEORSTACKTRACE`(iTracePriority, iStackTracePriority, ...) do { ; } while (0)
 - Print a trace statement with an optional stack trace to the log (debug builds only)*

Typedefs

- typedef `AAX_ETracePriorityHost` `AAX_ETracePriority`

15.55.2 Macro Definition Documentation

15.55.2.1 kAAX_Trace_Priority_None

```
#define kAAX_Trace_Priority_None AAX_eTracePriorityHost_None
```

15.55.2.2 kAAX_Trace_Priority_Critical

```
#define kAAX_Trace_Priority_Critical AAX_eTracePriorityHost_Critical
```

15.55.2.3 kAAX_Trace_Priority_High

```
#define kAAX_Trace_Priority_High AAX_eTracePriorityHost_High
```

15.55.2.4 kAAX_Trace_Priority_Normal

```
#define kAAX_Trace_Priority_Normal AAX_eTracePriorityHost_Normal
```

15.55.2.5 kAAX_Trace_Priority_Low

```
#define kAAX_Trace_Priority_Low AAX_eTracePriorityHost_Low
```

15.55.2.6 kAAX_Trace_Priority_Lowest

```
#define kAAX_Trace_Priority_Lowest AAX_eTracePriorityHost_Lowest
```

15.55.2.7 AAX_TRACE_RELEASE

```
#define AAX_TRACE_RELEASE(  
    iPriority,  
    ... )
```

Value:

```
{ \  
    AAX_CHostServices::Trace ( iPriority, __VA_ARGS__ ); \  
};
```

Print a trace statement to the log.

Use this macro to print a trace statement to the log file. This macro will be included in all builds of the plug-in.

Notes

- This macro is compatible with both host and embedded (AAX DSP) environments
- Subject to a total line limit of 256 chars

Usage Each invocation of this macro takes a trace priority and a `printf`-style logging string.

Because output from this macro will be enabled on end users' systems under certain tracing configurations, logs should always be formatted with some standard information to avoid confusion between logs from different plug-ins. This is the recommended formatting for AAX_TRACE_RELEASE logs:

```
[Manufacturer name] [Plug-in name] [Plug-in version][logging text (indented)]
```

For example:

```
AAX_TRACE_RELEASE(kAAX_Trace_Priority_Normal, "%s %s %s;\tMy float: %f, My C-string: %s",  
    "MyCompany", "MyPlugIn", "1.0.2", myFloat, myCString);
```

See also

[DigiTrace Guide](#)

15.55.2.8 AAX_STACKTRACE_RELEASE

```
#define AAX_STACKTRACE_RELEASE(  
    iPriority,  
    ... )
```

Value:

```
{ \  
    AAX_CHostServices::StackTrace ( iPriority, iPriority, __VA_ARGS__ ); \  
};
```

Print a stack trace statement to the log.

See also

[AAX_TRACE_RELEASE](#)

15.55.2.9 AAX_TRACEORSTACKTRACE_RELEASE

```
#define AAX_TRACEORSTACKTRACE_RELEASE (
    iTracePriority,
    iStackTracePriority,
    ... )
```

Value:

```
{ \
    AAX_CHostServices::StackTrace ( iTracePriority, iStackTracePriority, __VA_ARGS__ ); \
};
```

Print a trace statement with an optional stack trace to the log.

Parameters

in	<i>iTracePriority</i>	The log priority at which the trace statement will be printed
in	<i>iStackTracePriority</i>	The log priority at which the stack trace will be printed

See also

[AAX_TRACE_RELEASE](#)

15.55.2.10 AAX_ASSERT

```
#define AAX_ASSERT(
    condition )
```

Value:

```
{ \
    if( ! ( condition ) ) { \
        AAX_CHostServices::HandleAssertFailure( __FILE__, __LINE__, #condition,
(int32_t)AAX_eAssertFlags_Log ); \
    } \
};
```

Asserts that a condition is true and logs an error if the condition is false.

Notes

- This macro will be compiled out of release builds.
- This macro is compatible with bost host and embedded ([AAX DSP](#)) environments.

Usage Each invocation of this macro takes a single argument, which is interpreted as a `bool`.
[AAX_ASSERT](#)(desiredValue == variableUnderTest);

15.55.2.11 AAX_DEBUGASSERT

```
#define AAX_DEBUGASSERT(  
    condition ) do { ; } while (0)
```

Asserts that a condition is true and logs an error if the condition is false (debug plug-in builds only)

See also

[AAX_ASSERT](#)

15.55.2.12 AAX_TRACE

```
#define AAX_TRACE(  
    iPriority,  
    ... ) do { ; } while (0)
```

Print a trace statement to the log (debug plug-in builds only)

Use this macro to print a trace statement to the log file from debug builds of a plug-in.

Notes

- This macro will be compiled out of release builds
- This macro is compatible with bost host and embedded ([AAX DSP](#)) environments
- Subject to a total line limit of 256 chars

Usage Each invocation of this macro takes a trace priority and a `printf`-style logging string. For example:
`AAX_TRACE(kAAX_Trace_Priority_Normal, "My float: %f, My C-string: %s", myFloat, myCString);`

See also

[DigiTrace Guide](#)

15.55.2.13 AAX_STACKTRACE

```
#define AAX_STACKTRACE(  
    iPriority,  
    ... ) do { ; } while (0)
```

Print a stack trace statement to the log (debug builds only)

See also

[AAX_TRACE](#)

15.55.2.14 AAX_TRACEORSTACKTRACE

```
#define AAX_TRACEORSTACKTRACE(  
    iTracePriority,  
    iStackTracePriority,  
    ... ) do { ; } while (0)
```

Print a trace statement with an optional stack trace to the log (debug builds only)

Parameters

in	<i>iTracePriority</i>	The log priority at which the trace statement will be printed
in	<i>iStackTracePriority</i>	The log priority at which the stack trace will be printed

See also

[AAX_TRACE](#)

15.55.3 Typedef Documentation

15.55.3.1 AAX_ETracePriority

```
typedef AAX_ETracePriorityHost AAX_ETracePriority
```

15.56 AAX_Assert.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2015, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00055 #ifndef AAX_ASSERT_H
00056 #define AAX_ASSERT_H
00057
00058 #include "AAX_Enums.h"
00059
00060
00170 #ifdef _TMS320C6X // TI-only
00171
00172     #ifndef TI_SHELL_TRACING_H
00173     #include "TI_Shell_Tracing.h"
00174     #endif
00175
00176     typedef AAX_ETracePriorityDSP EAAX_Trace_Priority;
00177
00178     #define kAAX_Trace_Priority_None        AAX_eTracePriorityDSP_None
00179     #define kAAX_Trace_Priority_Critical    AAX_eTracePriorityDSP_High
00180     #define kAAX_Trace_Priority_High        AAX_eTracePriorityDSP_High
00181     #define kAAX_Trace_Priority_Normal      AAX_eTracePriorityDSP_Normal
00182     #define kAAX_Trace_Priority_Low         AAX_eTracePriorityDSP_Low
00183     #define kAAX_Trace_Priority_Lowest      AAX_eTracePriorityDSP_Low
00184
00185     //Note that the Message provided to AAX_TRACE must be a cons string available for indefinite time
00186     // because sending it to the host is done asynchronously.
00187     #define AAX_TRACE_RELEASE( ... ) TISHELLTRACE( __VA_ARGS__ )
00188
00189     //Stack traces not supported on TI - just log
00190     #define AAX_STACKTRACE_RELEASE( ... ) TISHELLTRACE( __VA_ARGS__ )
00191     #define AAX_TRACEORSTACKTRACE_RELEASE( iTracePriority, iStackTracePriority, ... ) TISHELLTRACE(
iTracePriority, __VA_ARGS__ )
00192
00193     #define _STRINGIFY(x) #x
00194     #define _TOSTRING(x) _STRINGIFY(x)
00195
00196     #define AAX_ASSERT( condition ) \
```

```

00197     { \
00198         if( ! (condition) ) _DoTrace( AAX_eTracePriorityDSP_Assert, \
00199             CAT(CAT( CAT(__FILE__, ":"), _TOSTRING(__LINE__) ) , CAT(" failed: ", #condition) ) );\
00200     }
00201
00202     #if defined(_DEBUG)
00203         #define AAX_DEBUGASSERT( condition ) AAX_ASSERT( condition )
00204         #define AAX_TRACE( ... ) AAX_TRACE_RELEASE( __VA_ARGS__ )
00205         #define AAX_STACKTRACE( ... ) AAX_STACKTRACE_RELEASE( __VA_ARGS__ )
00206         #define AAX_TRACEORSTACKTRACE( iTracePriority, iStackTracePriority, ... )
00207             AAX_TRACEORSTACKTRACE_RELEASE( iTracePriority, iStackTracePriority, __VA_ARGS__ )
00208     #else
00209         #define AAX_DEBUGASSERT( condition ) do { ; } while (0)
00210         #define AAX_TRACE( ... ) do { ; } while (0)
00211         #define AAX_STACKTRACE( ... ) do { ; } while (0)
00212         #define AAX_TRACEORSTACKTRACE( ... ) do { ; } while (0)
00213     #endif
00214
00215 #else // Host:
00216
00217     #ifndef AAX_CHOSTSERVICES_H
00218         #include "AAX_CHostServices.h"
00219     #endif
00220
00221     typedef AAX_ETracePriorityHost AAX_ETracePriority;
00222
00223     #define kAAX_Trace_Priority_None          AAX_eTracePriorityHost_None
00224     #define kAAX_Trace_Priority_Critical      AAX_eTracePriorityHost_Critical
00225     #define kAAX_Trace_Priority_High          AAX_eTracePriorityHost_High
00226     #define kAAX_Trace_Priority_Normal        AAX_eTracePriorityHost_Normal
00227     #define kAAX_Trace_Priority_Low           AAX_eTracePriorityHost_Low
00228     #define kAAX_Trace_Priority_Lowest        AAX_eTracePriorityHost_Lowest
00229
00230     //Note that the Message provided to AAX_TRACE must be a const string available for indefinite time
00231     // because sending it to the host is done asynchronously on TI
00232     #define AAX_TRACE_RELEASE( iPriority, ... ) \
00233     { \
00234         AAX_CHostServices::Trace ( iPriority, __VA_ARGS__ ); \
00235     };
00236
00237     #define AAX_STACKTRACE_RELEASE( iPriority, ... ) \
00238     { \
00239         AAX_CHostServices::StackTrace ( iPriority, iPriority, __VA_ARGS__ ); \
00240     };
00241
00242     #define AAX_TRACEORSTACKTRACE_RELEASE( iTracePriority, iStackTracePriority, ... ) \
00243     { \
00244         AAX_CHostServices::StackTrace ( iTracePriority, iStackTracePriority, __VA_ARGS__ ); \
00245     };
00246
00247     #if defined(_DEBUG)
00248
00249         #define AAX_ASSERT( condition ) \
00250         { \
00251             if( ! ( condition ) ) { \
00252                 AAX_CHostServices::HandleAssertFailure( __FILE__, __LINE__, #condition,
00253 (int32_t)AAX_eAssertFlags_Log | (int32_t)AAX_eAssertFlags_Dialog ); \
00254             } \
00255         };
00256
00257         #define AAX_DEBUGASSERT( condition ) \
00258         { \
00259             if( ! ( condition ) ) { \
00260                 AAX_CHostServices::HandleAssertFailure( __FILE__, __LINE__, #condition,
00261 (int32_t)AAX_eAssertFlags_Log | (int32_t)AAX_eAssertFlags_Dialog ); \
00262             } \
00263         };
00264
00265         #define AAX_TRACE( iPriority, ... ) AAX_TRACE_RELEASE( iPriority, __VA_ARGS__ )
00266         #define AAX_STACKTRACE( iPriority, ... ) AAX_STACKTRACE_RELEASE( iPriority, __VA_ARGS__ )
00267         #define AAX_TRACEORSTACKTRACE( iTracePriority, iStackTracePriority, ... )
00268             AAX_TRACEORSTACKTRACE_RELEASE( iTracePriority, iStackTracePriority, __VA_ARGS__ )
00269     #else
00270         #define AAX_ASSERT( condition ) \
00271         { \
00272             if( ! ( condition ) ) { \
00273                 AAX_CHostServices::HandleAssertFailure( __FILE__, __LINE__, #condition,
00274 (int32_t)AAX_eAssertFlags_Log ); \
00275             } \
00276         };
00277
00278         #define AAX_DEBUGASSERT( condition ) do { ; } while (0)
00279         #define AAX_TRACE( iPriority, ... ) do { ; } while (0)
00280         #define AAX_STACKTRACE( iPriority, ... ) do { ; } while (0)
00281         #define AAX_TRACEORSTACKTRACE( iTracePriority, iStackTracePriority, ... ) do { ; } while (0)

```

```

00279     #endif
00280
00281 #endif
00282
00283
00284 #endif // include guard
00285 // end of AAX_Assert.h

```

15.57 AAX_Atomic.h File Reference

```

#include "AAX.h"
#include <stdint.h>

```

15.57.1 Description

Atomic operation utilities.

Macros

- #define [AAX_ATOMIC_H_](#)

Functions

- [uint32_t AAX_CALLBACK AAX_Atomic_IncThenGet_32](#) (uint32_t &ioData)
Increments a 32-bit value and returns the result.
- [uint32_t AAX_CALLBACK AAX_Atomic_DecThenGet_32](#) (uint32_t &ioData)
Decrements a 32-bit value and returns the result.
- [uint32_t AAX_CALLBACK AAX_Atomic_Exchange_32](#) (volatile uint32_t &ioValue, uint32_t inExchange↔
Value)
Return the original value of ioValue and then set it to inExchangeValue.
- [uint64_t AAX_CALLBACK AAX_Atomic_Exchange_64](#) (volatile uint64_t &ioValue, uint64_t inExchange↔
Value)
Return the original value of ioValue and then set it to inExchangeValue.
- [template<typename TPointer >](#)
[TPointer *AAX_CALLBACK AAX_Atomic_Exchange_Pointer](#) (TPointer *&ioValue, TPointer *inExchange↔
Value)
Perform an exchange operation on a pointer value.
- [bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_32](#) (volatile uint32_t &ioValue, uint32_t in↔
CompareValue, uint32_t inExchangeValue)
Perform a compare and exchange operation on a 32-bit value.
- [bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_64](#) (volatile uint64_t &ioValue, uint64_t in↔
CompareValue, uint64_t inExchangeValue)
Perform a compare and exchange operation on a 64-bit value.
- [template<typename TPointer >](#)
[bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_Pointer](#) (TPointer *&ioValue, TPointer *in↔
CompareValue, TPointer *inExchangeValue)
Perform a compare and exchange operation on a pointer value.
- [template<typename TPointer >](#)
[TPointer *AAX_CALLBACK AAX_Atomic_Load_Pointer](#) (TPointer const *const volatile *inValue)
Atomically loads a pointer value.

15.57.2 Macro Definition Documentation

15.57.2.1 AAX_ATOMIC_H_

```
#define AAX_ATOMIC_H_
```

15.57.3 Function Documentation

15.57.3.1 AAX_Atomic_IncThenGet_32()

```
uint32_t AAX_CALLBACK AAX_Atomic_IncThenGet_32 (
    uint32_t & ioData )
```

Increments a 32-bit value and returns the result.

15.57.3.2 AAX_Atomic_DecThenGet_32()

```
uint32_t AAX_CALLBACK AAX_Atomic_DecThenGet_32 (
    uint32_t & ioData )
```

Decrements a 32-bit value and returns the result.

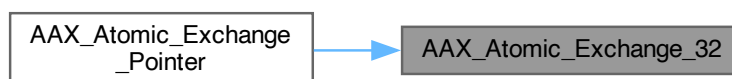
15.57.3.3 AAX_Atomic_Exchange_32()

```
uint32_t AAX_CALLBACK AAX_Atomic_Exchange_32 (
    volatile uint32_t & ioValue,
    uint32_t inExchangeValue )
```

Return the original value of ioValue and then set it to inExchangeValue.

Referenced by [AAX_Atomic_Exchange_Pointer\(\)](#).

Here is the caller graph for this function:



15.57.3.4 AAX_Atomic_Exchange_64()

```
uint64_t AAX_CALLBACK AAX_Atomic_Exchange_64 (
    volatile uint64_t & ioValue,
    uint64_t inExchangeValue )
```

Return the original value of ioValue and then set it to inExchangeValue.

Referenced by [AAX_Atomic_Exchange_Pointer\(\)](#).

Here is the caller graph for this function:



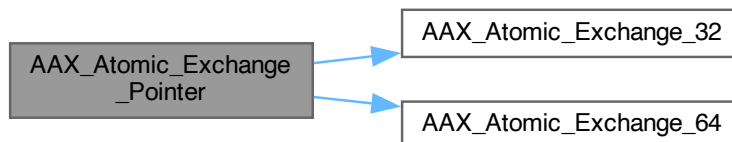
15.57.3.5 AAX_Atomic_Exchange_Pointer()

```
template<typename TPointer >
TPointer *AAX_CALLBACK AAX_Atomic_Exchange_Pointer (
    TPointer *& ioValue,
    TPointer * inExchangeValue )
```

Perform an exchange operation on a pointer value.

References [AAX_Atomic_Exchange_32\(\)](#), and [AAX_Atomic_Exchange_64\(\)](#).

Here is the call graph for this function:



15.57.3.6 AAX_Atomic_CompareAndExchange_32()

```
bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_32 (
    volatile uint32_t & ioValue,
    uint32_t inCompareValue,
    uint32_t inExchangeValue )
```

Perform a compare and exchange operation on a 32-bit value.

Referenced by [AAX_Atomic_CompareAndExchange_Pointer\(\)](#).

Here is the caller graph for this function:



15.57.3.7 AAX_Atomic_CompareAndExchange_64()

```
bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_64 (
    volatile uint64_t & ioValue,
    uint64_t inCompareValue,
    uint64_t inExchangeValue )
```

Perform a compare and exchange operation on a 64-bit value.

Referenced by [AAX_Atomic_CompareAndExchange_Pointer\(\)](#).

Here is the caller graph for this function:



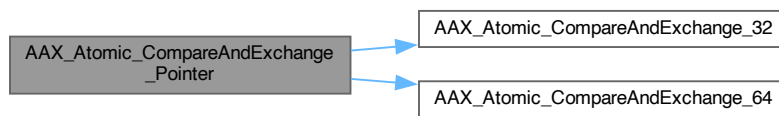
15.57.3.8 AAX_Atomic_CompareAndExchange_Pointer()

```
template<typename TPointer >
bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_Pointer (
    TPointer *& ioValue,
    TPointer * inCompareValue,
    TPointer * inExchangeValue )
```

Perform a compare and exchange operation on a pointer value.

References [AAX_Atomic_CompareAndExchange_32\(\)](#), and [AAX_Atomic_CompareAndExchange_64\(\)](#).

Here is the call graph for this function:



15.57.3.9 AAX_Atomic_Load_Pointer()

```
template<typename TPointer >
TPointer *AAX_CALLBACK AAX_Atomic_Load_Pointer (
    TPointer const *const volatile * inValue )
```

Atomically loads a pointer value.

15.58 AAX_Atomic.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2013-2015, 2018, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  *
00010  */
00011
00017 /*=====*/
00018
00019
00020 #pragma once
00021
00022 #ifndef AAX_ATOMIC_H_
00023 #define AAX_ATOMIC_H_
00024
00025 #include "AAX.h"
00026 #include <stdint.h>
00027
00028 #if (!defined AAX_PointerSize)
```

```

00029 #error Undefined pointer size
00030 #endif
00031
00032
00034 uint32_t AAX_CALLBACK AAX_Atomic_IncThenGet_32(uint32_t & ioData);
00035
00037 uint32_t AAX_CALLBACK AAX_Atomic_DecThenGet_32(uint32_t & ioData);
00038
00040 uint32_t AAX_CALLBACK AAX_Atomic_Exchange_32(
00041     volatile uint32_t& ioValue,
00042     uint32_t          inExchangeValue);
00043
00045 uint64_t AAX_CALLBACK AAX_Atomic_Exchange_64(
00046     volatile uint64_t& ioValue,
00047     uint64_t          inExchangeValue);
00048
00050 template<typename TPointer> TPointer* AAX_CALLBACK AAX_Atomic_Exchange_Pointer(TPointer*& ioValue,
    TPointer* inExchangeValue)
00051 {
00052     #if (AAX_PointerSize == AAXPointer_64bit)
00053         return (TPointer*)AAX_Atomic_Exchange_64(*(uint64_t*)(void*)&ioValue, (uint64_t)inExchangeValue);
00054     #elif (AAX_PointerSize == AAXPointer_32bit)
00055         return (TPointer*)AAX_Atomic_Exchange_32(*(uint32_t*)(void*)&ioValue, (uint32_t)inExchangeValue);
00056     #else
00057         #error Unsupported pointer size
00058     #endif
00059 }
00060
00062 bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_32(
00063     volatile uint32_t & ioValue,
00064     uint32_t          inCompareValue,
00065     uint32_t          inExchangeValue);
00066
00068 bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_64(
00069     volatile uint64_t& ioValue,
00070     uint64_t          inCompareValue,
00071     uint64_t          inExchangeValue);
00072
00074 template<typename TPointer> bool AAX_CALLBACK AAX_Atomic_CompareAndExchange_Pointer(TPointer*&
    ioValue, TPointer* inCompareValue, TPointer* inExchangeValue)
00075 {
00076     #if (AAX_PointerSize == AAXPointer_64bit)
00077         return AAX_Atomic_CompareAndExchange_64(*(uint64_t*)(void*)&ioValue, (uint64_t)inCompareValue,
            (uint64_t)inExchangeValue);
00078     #elif (AAX_PointerSize == AAXPointer_32bit)
00079         return AAX_Atomic_CompareAndExchange_32(*(uint32_t*)(void*)&ioValue, (uint32_t)inCompareValue,
            (uint32_t)inExchangeValue);
00080     #else
00081         #error Unsupported pointer size
00082     #endif
00083 }
00084
00086 template<typename TPointer> TPointer* AAX_CALLBACK AAX_Atomic_Load_Pointer(TPointer const * const
    volatile * inValue);
00087
00088
00089
00090 //TODO: Update all atomic function implementatons with proper acquire/release semantics
00091
00092
00093 // GCC/LLVM
00094 #if defined(__GNUC__)
00095
00096 // These intrinsics require GCC 4.2 or later
00097 #if (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 2))
00098
00099 inline uint32_t AAX_CALLBACK
00100 AAX_Atomic_IncThenGet_32(uint32_t& ioData)
00101 {
00102     return __sync_add_and_fetch(&ioData, 1);
00103 }
00104
00105 inline uint32_t AAX_CALLBACK
00106 AAX_Atomic_DecThenGet_32(uint32_t& ioData)
00107 {
00108     return __sync_sub_and_fetch(&ioData, 1);
00109 }
00110
00111 inline uint32_t
00112 AAX_Atomic_Exchange_32(
00113     volatile uint32_t& ioValue,
00114     uint32_t          inExchangeValue)
00115 {
00116     return __sync_lock_test_and_set(&ioValue, inExchangeValue);
00117 }
00118
00119 inline uint64_t

```

```

00120 AAX_Atomic_Exchange_64(
00121     volatile uint64_t& ioValue,
00122     uint64_t          inExchangeValue)
00123 {
00124     return __sync_lock_test_and_set(&ioValue, inExchangeValue);
00125 }
00126
00127 inline bool
00128 AAX_Atomic_CompareAndExchange_32(
00129     volatile uint32_t & ioValue,
00130     uint32_t          inCompareValue,
00131     uint32_t          inExchangeValue)
00132 {
00133     return __sync_bool_compare_and_swap(&ioValue, inCompareValue, inExchangeValue);
00134 }
00135
00136 inline bool
00137 AAX_Atomic_CompareAndExchange_64(
00138     volatile uint64_t& ioValue,
00139     uint64_t          inCompareValue,
00140     uint64_t          inExchangeValue)
00141 {
00142     return __sync_bool_compare_and_swap(&ioValue, inCompareValue, inExchangeValue);
00143 }
00144
00145 //TODO: Add GCC version check and alternative implementations for GCC versions that do not support
00146     __atomic_load
00147 template<typename TPointer>
00148 inline TPointer*
00149 AAX_Atomic_Load_Pointer(TPointer const * const volatile * inValue)
00150 {
00151     TPointer* value;
00152     __atomic_load(const_cast<TPointer * volatile *>(inValue), &(value), __ATOMIC_ACQUIRE);
00153     return value;
00154 }
00155 #else // (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 2))
00156 #error This file requires GCC 4.2 or later
00157 #endif // (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 2))
00158 // End GCC/LLVM
00159
00160
00161
00162
00163
00164 // Visual C
00165 #elif defined(_MSC_VER)
00166
00167 #ifndef __INTRIN_H_
00168 #include <intrin.h>
00169 #endif
00170
00171 #pragma intrinsic( _InterlockedIncrement, \
00172                  _InterlockedDecrement, \
00173                  _InterlockedExchange, \
00174                  _InterlockedCompareExchange, \
00175                  _InterlockedCompareExchange64)
00176
00177 inline uint32_t AAX_CALLBACK
00178 AAX_Atomic_IncThenGet_32(register uint32_t& ioData)
00179 {
00180     return static_cast<uint32_t>(_InterlockedIncrement((volatile long*)&ioData));
00181 }
00182
00183 inline uint32_t AAX_CALLBACK
00184 AAX_Atomic_DecThenGet_32(register uint32_t& ioData)
00185 {
00186     return static_cast<uint32_t>(_InterlockedDecrement((volatile long*)&ioData));
00187 }
00188
00189 inline uint32_t
00190 AAX_Atomic_Exchange_32(
00191     volatile uint32_t& ioDestination,
00192     uint32_t          inExchangeValue)
00193 {
00194     return static_cast<uint32_t>(_InterlockedExchange((volatile long*)&ioDestination,
00195 (long)inExchangeValue));
00196 }
00197
00198 #if (AAX_PointerSize == AAXPointer_64bit)
00199 #pragma intrinsic( _InterlockedExchange64, \
00200                  _InterlockedOr64)
00201
00202 inline uint64_t
00203 AAX_Atomic_Exchange_64(
00204     volatile uint64_t& ioValue,

```

```

00205     uint64_t          inExchangeValue)
00206 {
00207     return static_cast<uint64_t>(_InterlockedExchange64((volatile __int64*)&ioValue,
00208     (__int64)inExchangeValue));
00209 }
00210 template<typename TPointer>
00211 inline TPointer*
00212 AAX_Atomic_Load_Pointer(TPointer const * const volatile * inValue)
00213 {
00214     // Itanium supports acquire semantics
00215     #if (_M_IA64)
00216         return reinterpret_cast<TPointer*>(_InterlockedOr64_acq(const_cast<__int64 volatile
00217         *>(reinterpret_cast<const __int64 volatile *>(inValue)), 0x0000000000000000));
00218     #else
00219         return reinterpret_cast<TPointer*>(_InterlockedOr64(const_cast<__int64 volatile
00220         *>(reinterpret_cast<const __int64 volatile *>(inValue)), 0x0000000000000000));
00221     #endif
00222 }
00223 #elif (AAX_PointerSize == AAXPointer_32bit)
00224 #pragma intrinsic( _InterlockedOr )
00225 // _InterlockedExchange64 is not available on 32-bit Pentium in Visual C
00226 inline uint64_t
00227 AAX_Atomic_Exchange_64(
00228     volatile uint64_t& ioValue,
00229     uint64_t          inExchangeValue)
00230 {
00231     for(;;)
00232     {
00233         uint64_t result = ioValue;
00234         if(AAX_Atomic_CompareAndExchange_64(ioValue, result, inExchangeValue))
00235         {
00236             return result;
00237         }
00238     }
00239 }
00240 return 0; // will never get here
00241 }
00242 }
00243 template<typename TPointer>
00244 inline TPointer*
00245 AAX_Atomic_Load_Pointer(TPointer const * const volatile * inValue)
00246 {
00247     return reinterpret_cast<TPointer*>(_InterlockedOr(const_cast<long volatile
00248     *>(reinterpret_cast<const long volatile *>(inValue)), 0x00000000));
00249 }
00250 #endif
00251 inline bool
00252 AAX_Atomic_CompareAndExchange_32(
00253     uint32_t volatile & ioValue,
00254     uint32_t          inCompareValue,
00255     uint32_t          inExchangeValue)
00256 {
00257     return static_cast<uint32_t>(_InterlockedCompareExchange((volatile long*)&ioValue,
00258     (long)inExchangeValue, (long)inCompareValue)) == inCompareValue;
00259 }
00260 inline bool
00261 AAX_Atomic_CompareAndExchange_64(
00262     volatile uint64_t& ioValue,
00263     uint64_t          inCompareValue,
00264     uint64_t          inExchangeValue)
00265 {
00266     return static_cast<uint64_t>(_InterlockedCompareExchange64((volatile __int64*)&ioValue,
00267     (__int64)inExchangeValue, (__int64)inCompareValue)) == inCompareValue;
00268 }
00269 // End Visual C
00270 #else // Not Visual C or GCC/LLVM
00271 #error Provide an atomic operation implementation for this compiler
00272 #endif // Compiler version check
00273 #endif // #ifndef AAX_ATOMIC_H_

```

15.59 AAX_Callbacks.h File Reference

```
#include "AAX.h"
```

15.59.1 Description

AAX callback prototypes and ProcPtr definitions

Classes

- class [AAX_Component< aContextType >](#)
Empty class containing type declarations for the AAX algorithm and associated callbacks.

Typedefs

- typedef [IACFUnknown](#) *([AAX_CALLBACK](#) * [AAXCreateObjectProc](#)) (void)
- typedef [AAX_Component](#)< void >::CProcessProc [AAX_CProcessProc](#)
A user-defined callback that AAX calls to process data packets and/or audio.
- typedef [AAX_Component](#)< void >::CPacketAllocator [AAX_CPacketAllocator](#)
Used by [AAX_SchedulePacket\(\)](#)
- typedef [AAX_Component](#)< void >::CInstanceInitProc [AAX_CInstanceInitProc](#)
A user-defined callback that AAX calls to notify the component that an instance is being added or removed.
- typedef [AAX_Component](#)< void >::CBackgroundProc [AAX_CBackgroundProc](#)
A user-defined callback that AAX calls in the AAX Idle time.
- typedef [AAX_Component](#)< void >::CInitPrivateDataProc [AAX_CInitPrivateDataProc](#)
A user-defined callback to initialize a private data block.

Enumerations

- enum [AAX_CProcPtrID](#) {
 [kAAX_ProcPtrID_Create_EffectParameters](#) = 0 ,
 [kAAX_ProcPtrID_Create_EffectGUI](#) = 1 ,
 [kAAX_ProcPtrID_Create_HostProcessor](#) = 3 ,
 [kAAX_ProcPtrID_Create_EffectDirectData](#) = 5 ,
 [kAAX_ProcPtrID_Create_TaskAgent](#) = 6 ,
 [kAAX_ProcPtrID_Create_SessionDocumentClient](#) = 7 }

15.59.2 Typedef Documentation

15.59.2.1 AAXCreateObjectProc

```
typedef IACFUnknown *(AAX\_CALLBACK * AAXCreateObjectProc) (void)
```

15.59.2.2 AAX_CProcessProc

```
typedef AAX_Component<void>::CProcessProc AAX_CProcessProc
```

A user-defined callback that AAX calls to process data packets and/or audio.

iContextPtrsBegin

A vector of context pointers. Each element points to the context for one instance of this component. `iContextPtrsEnd` gives the upper bound of the vector and `(iContextPtrsEnd - iContextPtrsBegin)` gives the count.

iContextPtrsEnd

The upper bound of the vector at `iContextPtrsBegin`. `(iContextPtrsEnd - iContextPtrsBegin)` gives the count of this vector.

The instance vector was originally NULL-terminated in earlier versions of this API. However, the STL-style begin/end pattern was suggested as a more general representation that could, for instance, allow a vector to be split for parallel processing.

15.59.2.3 AAX_CPacketAllocator

```
typedef AAX_Component<void>::CPacketAllocator AAX_CPacketAllocator
```

Used by `AAX_SchedulePacket()`

Deprecated

A `AAX_CProcessProc` that calls `AAX_SchedulePacket()` must include a `AAX_CPacketAllocator` field in its context and register that field with `AAX`. `AAX` will then populate that field with a `AAX_CPacketAllocator` to pass to `AAX_SchedulePacket()`.

See also

`AAX_SchedulePacket()`

15.59.2.4 AAX_CInstanceInitProc

```
typedef AAX_Component<void>::CInstanceInitProc AAX_CInstanceInitProc
```

A user-defined callback that AAX calls to notify the component that an instance is being added or removed.

This optional callback allows the component to keep per-instance data. It's called before the instance appears in the list supplied to `CProcessProc`, and then after the instance is removed from the list.

iInstanceContextPtr

A pointer to the context data structure of the instance being added or removed from the processing list.

iAction

Indicates the action that triggered the init callback, e.g. whether the instance is being added or removed.

Return values

<i>Should</i>	return 0 on success, anything else on failure. Failure will prevent the instance from being created.
---------------	--

15.59.2.5 AAX_CBackgroundProc

```
typedef AAX_Component<void>::CBackgroundProc AAX_CBackgroundProc
```

A user-defined callback that AAX calls in the AAX Idle time.

This optional callback allows the component to do background processing in whatever manner the plug-in developer desires

Return values

<i>Should</i>	return 0 on success, anything else on failure. Failure will cause the AAX host to signal an error up the callchain.
---------------	---

15.59.2.6 AAX_CInitPrivateDataProc

```
typedef AAX_Component<void>::CInitPrivateDataProc AAX_CInitPrivateDataProc
```

A user-defined callback to initialize a private data block.

Deprecated

A component that requires private data supplies [AAX_CInitPrivateDataProc](#) callbacks to set its private data to the state it should be in at the start of audio. The component first declares one or more pointers to private data in its context. It then registers each such field with AAX along with its data size, various other attributes, and a [AAX_CInitPrivateDataProc](#). The [AAX_CInitPrivateDataProc](#) always runs on the host system, not the DSP. AAX allocates storage for each private data block and calls its associated [AAX_CInitPrivateDataProc](#) to initialize it. If the component's [AAX_CProcessProc](#) runs on external hardware, AAX initializes private data blocks on the host system and copies them to the remote system.

See also

`alg_pd_init`

inFieldIndex

The port ID of the block to be initialized, as generated by [AAX_FIELD_INDEX\(\)](#). A component can register a separate [AAX_CInitPrivateDataProc](#) for each of its private data blocks, or it can use fewer functions that switch on `inFieldIndex`.

inNewBlock

A pointer to the block to be initialized. If the component runs externally, AAX will copy this block to the remote system after it is initialized.

inSize

The size of the block to be initialized. If a component has multiple private blocks that only need to be zeroed out, say, it can use a single `AAX_CInitPrivateDataProc` for all of these blocks that zeroes them out according to `inSize`.

inController

A pointer to the current Effect instance's [AAX_IController](#).

Note

Do not directly reference data from this interface when populating `iNewBlock`. The data in this block must be fully self-contained to ensure portability to a new device or memory space.

Deprecated**15.59.3 Enumeration Type Documentation****15.59.3.1 AAX_CProcPtrID**

enum [AAX_CProcPtrID](#)

Enumerator

<code>kAAX_ProcPtrID_Create_EffectParameters</code>	AAX_IEffectParameters creation procedure
<code>kAAX_ProcPtrID_Create_EffectGUI</code>	AAX_IEffectGUI creation procedure
<code>kAAX_ProcPtrID_Create_HostProcessor</code>	AAX_IHostProcessor creation procedure
<code>kAAX_ProcPtrID_Create_EffectDirectData</code>	AAX_IEffectDirectData creation procedure, used by plug-ins that want direct access to their alg memory
<code>kAAX_ProcPtrID_Create_TaskAgent</code>	AAX_ITaskAgent creation procedure, used by plug-ins that want to process task requests made by the host.
<code>kAAX_ProcPtrID_Create_SessionDocumentClient</code>	AAX_ISessionDocumentClient creation procedure

15.60 AAX_Callbacks.h

[Go to the documentation of this file.](#)

```
00001  /*=====*/
00002  /*
00003  *
```



```

00004  * Copyright 2014-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019  /*=====*/
00020
00021
00023 #ifndef AAX_CALLBACKS_H_
00024 #define AAX_CALLBACKS_H_
00026
00027 #include "AAX.h"
00028
00029 // Callback IDs
00030 enum AAX_CProcPtrID
00031 {
00032     kAAX_ProcPtrID_Create_EffectParameters = 0,
00033     kAAX_ProcPtrID_Create_EffectGUI = 1,
00034     kAAX_ProcPtrID_Create_HostProcessor = 3,
00035     kAAX_ProcPtrID_Create_EffectDirectData = 5,
00036     kAAX_ProcPtrID_Create_TaskAgent = 6,
00037     kAAX_ProcPtrID_Create_SessionDocumentClient = 7
00038 };
00039
00040 class IACFUnknown;
00041
00042 typedef IACFUnknown* (AAX_CALLBACK *AAXCreateObjectProc) (void);
00043
00044
00048 template <typename aContextType>
00049 class AAX_Component
00050 {
00051     public:
00052
00053     typedef void
00054         (AAX_CALLBACK *CProcessProc) (
00055             aContextType * const inContextPtrsBegin [],
00056             const void * inContextPtrsEnd);
00057
00058     typedef void *
00059         (AAX_CALLBACK *CPacketAllocator) (
00060             const aContextType * inContextPtr,
00061             AAX_CFieldIndex inOutputPort,
00062             AAX_CTimestamp inTimestamp);
00063
00064     typedef int32_t
00065         (AAX_CALLBACK *CInstanceInitProc) (
00066             const aContextType * inInstanceContextPtr,
00067             AAX_EComponentInstanceInitAction iAction );
00068
00069     typedef int32_t
00070         (AAX_CALLBACK *CBackgroundProc) ( void );
00071
00072     typedef void
00073         (AAX_CALLBACK *CInitPrivateDataProc) (
00074             AAX_CFieldIndex inFieldIndex,
00075             void * inNewBlock,
00076             int32_t inSize,
00077             IACFUnknown * const inController);
00078
00079 };
00080
00101 typedef AAX_Component <void>::CProcessProc AAX_CProcessProc;
00102
00115 typedef AAX_Component <void>::CPacketAllocator AAX_CPacketAllocator;
00116
00137 typedef AAX_Component <void>::CInstanceInitProc AAX_CInstanceInitProc;
00138
00148 typedef AAX_Component <void>::CBackgroundProc AAX_CBackgroundProc;
00149
00191 typedef AAX_Component <void>::CInitPrivateDataProc AAX_CInitPrivateDataProc;
00192
00194 #endif // AAX_CALLBACKS_H_

```

15.61 AAX_CArrayDataBuffer.h File Reference

```

#include "AAX_IDataBuffer.h"
#include "AAX.h"

```

```
#include <string>
#include <limits>
#include <type_traits>
```

Classes

- class [AAX_CArrayDataBufferOfType< T, D >](#)
A convenience class for array data buffers.
- class [AAX_CArrayDataBuffer< D >](#)

Macros

- #define [AAX_CArrayDataBuffer_H](#)

15.61.1 Macro Definition Documentation

15.61.1.1 AAX_CArrayDataBuffer_H

```
#define AAX_CArrayDataBuffer_H
```

15.62 AAX_CArrayDataBuffer.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019
00020 #ifndef AAX_CArrayDataBuffer_H
00021 #define AAX_CArrayDataBuffer_H
00022
00023 #include "AAX_IDataBuffer.h"
00024 #include "AAX.h"
00025
00026 #include <string>
00027 #include <limits>
00028 #include <type_traits>
00029
00030
00036 template <AAX_CTypeID T, class D>
00037 class AAX_CArrayDataBufferOfType : public AAX_IDataBuffer
00038 {
00039 public:
00040     explicit AAX_CArrayDataBufferOfType (std::vector<D> const & inData) : mData{inData} {}
00041     explicit AAX_CArrayDataBufferOfType (std::vector<D> && inData) : mData{inData} {}
00042
00043     AAX_CArrayDataBufferOfType(AAX_CArrayDataBufferOfType const &) = default;
```

```

00044     AAX_CArrayDataBufferOfType(AAX_CArrayDataBufferOfType &&) = default;
00045
00046     ~AAX_CArrayDataBufferOfType (void) AAX_OVERRIDE = default;
00047
00048     AAX_CArrayDataBufferOfType& operator= (AAX_CArrayDataBufferOfType const & other) = default;
00049     AAX_CArrayDataBufferOfType& operator= (AAX_CArrayDataBufferOfType && other) = default;
00050
00051     AAX_Result Type(AAX_CTypeID * oType) const AAX_OVERRIDE {
00052         if (!oType) { return AAX_ERROR_NULL_ARGUMENT; }
00053         *oType = T;
00054         return AAX_SUCCESS;
00055     }
00056     AAX_Result Size(int32_t * oSize) const AAX_OVERRIDE {
00057         if (!oSize) { return AAX_ERROR_NULL_ARGUMENT; }
00058         auto const size = mData.size() * sizeof(D);
00059         static_assert(std::numeric_limits<decltype(size)>::max() >=
std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max(),
        "size variable may not represent all positive values of oSize");
00061         if (size > std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max()) {
00062             return AAX_ERROR_SIGNED_INT_OVERFLOW;
00063         }
00064         *oSize = static_cast<std::remove_pointer<decltype(oSize)>::type>(size);
00065         return AAX_SUCCESS;
00066     }
00067     AAX_Result Data(void const ** oBuffer) const AAX_OVERRIDE {
00068         if (!oBuffer) { return AAX_ERROR_NULL_ARGUMENT; }
00069         *oBuffer = mData.data();
00070         return AAX_SUCCESS;
00071     }
00072 private:
00073     std::vector<D> const mData;
00074 };
00075
00076 template <class D>
00077 class AAX_CArrayDataBuffer : public AAX_IDataBuffer
00078 {
00079 public:
00080     AAX_CArrayDataBuffer (AAX_CTypeID inType, std::vector<D> const & inData) : mType{inType},
mData{inData} {}
00084     AAX_CArrayDataBuffer (AAX_CTypeID inType, std::vector<D> && inData) : mType{inType}, mData{inData}
{}
00085
00086     AAX_CArrayDataBuffer(AAX_CArrayDataBuffer const &) = default;
00087     AAX_CArrayDataBuffer(AAX_CArrayDataBuffer &&) = default;
00088
00089     ~AAX_CArrayDataBuffer (void) AAX_OVERRIDE = default;
00090
00091     AAX_CArrayDataBuffer& operator= (AAX_CArrayDataBuffer const & other) = default;
00092     AAX_CArrayDataBuffer& operator= (AAX_CArrayDataBuffer && other) = default;
00093
00094     AAX_Result Type(AAX_CTypeID * oType) const AAX_OVERRIDE {
00095         if (!oType) { return AAX_ERROR_NULL_ARGUMENT; }
00096         *oType = mType;
00097         return AAX_SUCCESS;
00098     }
00099     AAX_Result Size(int32_t * oSize) const AAX_OVERRIDE {
00100         if (!oSize) { return AAX_ERROR_NULL_ARGUMENT; }
00101         auto const size = mData.size() * sizeof(D);
00102         static_assert(std::numeric_limits<decltype(size)>::max() >=
std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max(),
        "size variable may not represent all positive values of oSize");
00104         if (size > std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max()) {
00105             return AAX_ERROR_SIGNED_INT_OVERFLOW;
00106         }
00107         *oSize = static_cast<std::remove_pointer<decltype(oSize)>::type>(size);
00108         return AAX_SUCCESS;
00109     }
00110     AAX_Result Data(void const ** oBuffer) const AAX_OVERRIDE {
00111         if (!oBuffer) { return AAX_ERROR_NULL_ARGUMENT; }
00112         *oBuffer = mData.data();
00113         return AAX_SUCCESS;
00114     }
00115 private:
00116     AAX_CTypeID const mType;
00117     std::vector<D> const mData;
00118 };
00119
00120 #endif

```

15.63 AAX_CAtomicQueue.h File Reference

```
#include "AAX_IPointerQueue.h"
#include "AAX_Atomic.h"
#include "AAX_CMutex.h"
#include <cstring>
```

15.63.1 Description

Atomic, non-blocking queue.

Classes

- class [AAX_CAtomicQueue< T, S >](#)

15.64 AAX_CAtomicQueue.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  *
00010  */
00011
00018 /*=====*/
00020 #ifndef AAX_CATOMICQUEUE_H
00021 #define AAX_CATOMICQUEUE_H
00023
00024
00025 // AAX Includes
00026 #include "AAX_IPointerQueue.h"
00027 #include "AAX_Atomic.h"
00028 #include "AAX_CMutex.h"
00029
00030 // Standard Includes
00031 #include <cstring>
00032
00033
00051 template <typename T, size_t S>
00052 class AAX_CAtomicQueue : public AAX_IPointerQueue<T>
00053 {
00054 public:
00055     virtual ~AAX_CAtomicQueue() {}
00056     AAX_CAtomicQueue();
00057
00058 public:
00059     static const size_t template_size = S;
00060
00061     typedef typename AAX_IPointerQueue<T>::template_type template_type;
00062     typedef typename AAX_IPointerQueue<T>::value_type value_type;
00063
00064 public: // AAX_IContainer
00065     virtual void Clear();
00066
00067 public: // AAX_IPointerQueue
00068     virtual AAX_IContainer::EStatus Push(value_type inElem);
00069     virtual value_type Pop();
00070     virtual value_type Peek() const;
00071
00072 private:
00073     AAX_CMutex mMutex;
00074     uint32_t mReadIdx;
```

```

00075     uint32_t mWriteIdx;
00076     value_type mRingBuffer[S];
00077 };
00078
00079
00081
00082 template <typename T, size_t S>
00083 inline AAX_CAtomicQueue<T, S>::AAX_CAtomicQueue()
00084 : AAX_IPointerQueue<T>()
00085 , mMutex()
00086 , mReadIdx(0)
00087 , mWriteIdx(0)
00088 {
00089     Clear();
00090 }
00091
00092 template <typename T, size_t S>
00093 inline void AAX_CAtomicQueue<T, S>::Clear()
00094 {
00095     std::memset((void*)mRingBuffer, 0x0, sizeof(mRingBuffer));
00096 }
00097
00098 template <typename T, size_t S>
00099 inline AAX_IContainer::EStatus AAX_CAtomicQueue<T, S>::Push(typename
AAX_CAtomicQueue<T, S>::value_type inElem)
00100 {
00101     if (NULL == inElem)
00102     {
00103         return AAX_IContainer::eStatus_Unsupported;
00104     }
00105
00106     AAX_IContainer::EStatus result = AAX_IContainer::eStatus_Unavailable;
00107
00108     AAX_StLock_Guard guard(mMutex);
00109     //
00110     // Possible failure case without mutex is because of several write threads try to modify
00111     // mWriteIdx concurrently
00112     //
00113     // Example:
00114     //
00115     // -
00116     // Notation:
00117     // First number - write thread number
00118     // Second number - value number
00119     // 1/15 - 1st thread that write number 15
00120     //
00121     // -
00122     // Queue may look like this:
00123     //           mReadIdx
00124     //           |
00125     // |.....| 4/3 | 4/4 | 1/4 | 1/5 | 2/7 | 2/8 | 2/9 | .....|
00126     //           |
00127     //           mWriteIdx
00128     // place# | 0 | 1 | 2 | 3 | 4 | 5 | 6 | .....|
00129     //
00130     // -
00131     // Possible operation order (w stands for mWriteIdx, r - for mReadIdx):
00132     //-----
00133     // thread#| action | write index value | mWriteIdx |
00134     //         |         | internal variable |           |
00135     //-----
00136     //      5 |  w++ |         2 |         2 |
00137     //-----
00138     //      6 |  w++ |         3 |         3 |
00139     //-----
00140     //      5 | false |         - | 2not=3 => 2--=1 |
00141     //-----
00142     //   read |  r++ |         - |         - |
00143     //-----
00144     //   read |  r++ |         - |         - |
00145     //-----
00146     // -
00147     // Queue state:
00148     //           mReadIdx
00149     //           |
00150     // |.....| 4/3 | 0 | 0 | 1/5 | 2/7 | 2/8 | 2/9 | .....|
00151     //           |
00152     //           mWriteIdx
00153     // place# | 0 | 1 | 2 | 3 | 4 | 5 | 6 | .....|
00154     //
00155     // -
00156     //-----
00157     //      6 | false |         - | 3not=1 => 3--=2 | // place 3 is still not empty to write
00158     //-----
00159     //
00160     // -
00161     // Now, some other thread (5, for example) can successfully write

```

```

00162 // it's value to queue and move mWriteIdx forward:
00163 //
00164 // -
00165 // Queue state:
00166 //           mReadIdx
00167 //           |
00168 // |.....| 4/3 | 0 | 5/1 | 1/5 | 2/7 | 2/8 | 2/9 | .....|
00169 //           |
00170 //           mWriteIdx
00171 // place# | 0 | 1 | 2 | 3 | 4 | 5 | 6 | .....|
00172 //
00173 // -
00174 // Thus, we have one place with NULL value left. In the next round mReadIdx will
00175 // stuck on place #1 (queue thinks that it's empty) until one of the write threads
00176 // will write the value into place #1. It could be thread #5, so we have:
00177 //
00178 // -
00179 // Queue state:
00180 //           mReadIdx
00181 //           |
00182 // |.....| 9/1 | 5/9 | 5/1 | 1/5 | 2/7 | 2/8 | 2/9 | .....|
00183 //           |
00184 //           mWriteIdx
00185 // place# | 0 | 1 | 2 | 3 | 4 | 5 | 6 | .....|
00186 //
00187 // -
00188 // And we will read 5/9 before 5/1
00189 //
00190 //
00191 // Note that read/write both begin at index 1
00192 const uint32_t idx = AAX_Atomic_IncThenGet_32(mWriteIdx);
00193 const uint32_t widx = idx % S;
00194
00195 // Do the push. If the value at the current write index is non-NULL then we have filled the
buffer.
00196 const bool cxResult = AAX_Atomic_CompareAndExchange_Pointer(mRingBuffer[widx], (value_type)0x0,
inElem);
00197
00198 if (false == cxResult)
00199 {
00200     result = AAX_IContainer::eStatus_Overflow;
00201
00202     const uint32_t ridx = (0 == idx) ? S : idx-1;
00203
00204     // Note the write index has already been incremented, so in the event of an overflow we must
00205     // return the write index to its previous location.
00206     //
00207     // Note: if multiple write threads encounter concurrent push overflows then the write pointer
00208     // will not be fully decremented back to the overflow location, and the read index will need
00209     // to increment multiple positions to clear the overflow state.
00210     const bool resetResult = AAX_Atomic_CompareAndExchange_32(mWriteIdx, idx, ridx);
00211     AAX_Atomic_CompareAndExchange_32(mWriteIdx, idx, ridx);
00212
00213     printf("AAX_CAtomicQueue: overflow - reset: %s, idx: %lu, widx: %lu, inElem: %p\n",
00214         resetResult ? "yes" : "no",
00215         (unsigned long)idx,
00216         (unsigned long)widx,
00217         inElem);
00218 }
00219 else
00220 {
00221     result = AAX_IContainer::eStatus_Success;
00222
00223     // Handle wraparound
00224     //
00225     // There may be multiple write threads pushing elements at the same time, so we use
00226     // (wrapped index < raw index) instead of (raw index == boundary)
00227     //
00228     // This assumes overhead between S and UINT_32_MAX of at least as many elements as
00229     // there are write threads.
00230
00231     bool exchResult = false;
00232     if (widx < idx)
00233     {
00234         exchResult =
00235             AAX_Atomic_CompareAndExchange_32(mWriteIdx, idx, widx);
00236     }
00237
00238     printf("AAX_CAtomicQueue: pushed - reset: %s, idx: %lu, widx: %lu, inElem: %p\n",
00239         (widx < idx) ? exchResult ? "yes" : "no" : "n/a",
00240         (unsigned long)idx,
00241         (unsigned long)widx,
00242         inElem);
00243 }
00244
00245 return result;
00246 }

```

```

00247
00248 template <typename T, size_t S>
00249 inline typename AAX_CAtomicQueue<T, S>::value_type AAX_CAtomicQueue<T, S>::Pop()
00250 {
00251     // Note that read/write both begin at index 1
00252     mReadIdx = (mReadIdx+1) % template_size;
00253     value_type const val = AAX_Atomic_Exchange_Pointer(mRingBuffer[mReadIdx], (value_type)0x0);
00254
00255     // printf("AAX_CAtomicQueue: popped - reset: %s, idx: %lu, val: %p\n",
00256     //         (0x0 == val) ? "yes" : "no",
00257     //         (unsigned long)mReadIdx,
00258     //         val);
00259
00260     if (0x0 == val)
00261     {
00262         // If the value is NULL then no value has yet been written to this location. Decrement the
00263         read index
00264         --mReadIdx; // No need to handle wraparound since the read index will be incremented before
00265         the next read
00266     }
00267     return val;
00268 }
00269 template <typename T, size_t S>
00270 inline typename AAX_CAtomicQueue<T, S>::value_type AAX_CAtomicQueue<T, S>::Peek() const
00271 {
00272     // I don't think that we need a memory barrier here because:
00273     // a) mReadIdx will only be modified from the read thread, and therefore presumably
00274     //     using the same CPU (or at least I can't see any way for mReadIdx modification
00275     //     ordering to be a problem between Peek() and Pop() on a single thread.)
00276     // b) We don't care if mRingBuffer modifications are run out of order between the read
00277     //     and write threads, as long as they are "close".
00278     const uint32_t testIdx = (mReadIdx+1) % template_size;
00279     return AAX_Atomic_Load_Pointer(&mRingBuffer[testIdx]);
00280 }
00281
00282 // Attempt to support multiple read threads
00283 //
00284 // This approach is broken in the following scenario:
00285 //
00286 // Thread | Operation
00287 // -----|-----
00288 //      A | Pop v enter
00289 //      A | Pop - increment/get read index (get 1)
00290 //      A | Pop - exchange pointer (get 0x0)
00291 //   other | Push ptr1
00292 //   other | Push ptr2
00293 //      B | Pop v enter
00294 //      B | Pop - increment/get read index (get 2)
00295 //      B | Pop - exchange pointer (get ptr2)
00296 //      B | ERROR: popped ptr2 before ptr1
00297 //      B | Pop ^ exit
00298 //      A | Pop - decrement read index (set 1)
00299 //      A | Pop ^ exit
00300 //   any  | Pop v enter
00301 //   any  | Pop - increment/get read index (get 2)
00302 //   any  | Pop - exchange pointer (get 0x0)
00303 //   any  | ERROR: should be ptr2
00304 //   any  | This NULL state continues for further Pop calls until either Push wraps around
00305 //   any  | or another pair of concurrent calls to Pop just happens to re-align the read
00306 //   any  | index by incrementing twice before any reads occur
00307 //   any  | Pop - decrement read index (set 1)
00308 //   any  | Pop ^ exit
00309 //
00310 // This could be fixed by incrementing the read index until either a non-NULL value is found or
00311 // the initial position is reached, but that would have terrible performance.
00312 //
00313 // In any case, assuming a single read thread is optimal when we want maximum performance for read
00314 // operations, since this requires the fewest number of atomic operations in the read methods
00315 /*
00316 template <typename T, size_t S>
00317 inline typename AAX_CAtomicQueue<T, S>::value_type AAX_CAtomicQueue<T, S>::Pop()
00318 {
00319     const uint32_t idx = AAX_Atomic_IncThenGet_32(mReadIdx);
00320     const uint32_t widx = idx % S;
00321
00322     value_type const val = AAX_Atomic_Exchange_Pointer(mRingBuffer[widx], (value_type)0x0);
00323
00324     if (0x0 == val)
00325     {
00326         // If the value is NULL then no value has yet been written to this location. Decrement the
00327         read index
00328         AAX_Atomic_DecThenGet_32(mReadIdx);
00329     }
00330     else
00331     {
00332         // Handle wraparound (assumes some overhead between S and UINT_32_MAX)

```

```

00331         if (widx < idx)
00332         {
00333             AAX_Atomic_CompareAndExchange_32(mReadIdx, idx, widx);
00334         }
00335     }
00336
00337     return val;
00338 }
00339 */
00340
00342
00344 #endif /* defined(AAX_CATOMICQUEUE_H) */

```

15.65 AAX_CAutoreleasePool.h File Reference

15.65.1 Description

Autorelease pool helper utility.

Classes

- class [AAX_CAutoreleasePool](#)

Macros

- #define [_AAX_CAUTORELEASEPOOL_H_](#)

15.65.2 Macro Definition Documentation

15.65.2.1 _AAX_CAUTORELEASEPOOL_H_

```
#define _AAX_CAUTORELEASEPOOL_H_
```

15.66 AAX_CAutoreleasePool.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2014-2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  *
00010  */
00011 */
00012
00018 /*=====*/
00019
00020
00021 #pragma once
00022
00023 #ifndef _AAX_CAUTORELEASEPOOL_H_

```



```

00024 #define _AAX_CAUTORELEASEPOOL_H_
00025
00026
00027 /* \brief Creates an autorelease pool for the scope of the stack based class
00028    to clean up any autoreleased memory that was allocated in the lifetime of
00029    the pool.
00030
00031    \details
00032    This may be used on either Mac or Windows platforms and will not pull in
00033    any Cocoa dependencies.
00034
00035    usage:
00036    \code
00037    {
00038        AAX_CAutoreleasePool myAutoReleasePool
00039        delete myCocoaObject;
00040
00041        // Pool is released when the AAX_CAutoreleasePool is destroyed
00042    }
00043    \endcode
00044    */
00045 class AAX_CAutoreleasePool
00046 {
00047     public:
00048         AAX_CAutoreleasePool();
00049         ~AAX_CAutoreleasePool();
00050
00051     private:
00052         AAX_CAutoreleasePool (const AAX_CAutoreleasePool&);
00053         AAX_CAutoreleasePool& operator= (const AAX_CAutoreleasePool&);
00054
00055     private:
00056         void* mAutoreleasePool;
00057 };
00058
00059
00060 #endif // #ifndef _AAX_CAUTORELEASEPOOL_H_

```

15.67 AAX_CBinaryDisplayDelegate.h File Reference

```

#include "AAX_IDisplayDelegate.h"
#include "AAX_CString.h"
#include <vector>
#include <algorithm>

```

15.67.1 Description

A binary display delegate.

Classes

- class [AAX_CBinaryDisplayDelegate< T >](#)
A binary display format conforming to [AAX_IDisplayDelegate](#).

15.68 AAX_CBinaryDisplayDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013–2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.

```

```

00006  *
00007  *  CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  *  not disclose to any third party. Use of the information contained in this
00009  *  document is subject to an Avid SDK license.
00010  *
00011  */
00012
00013 /*=====*/
00014
00015 #ifndef AAX_CBINARYDISPLAYDELEGATE_H
00016 #define AAX_CBINARYDISPLAYDELEGATE_H
00017
00018 #include "AAX_IDisplayDelegate.h"
00019 #include "AAX_CString.h"
00020
00021 #include <vector>
00022 #ifdef WINDOWS_VERSION
00023 #include <algorithm>
00024 #endif
00025 #include <algorithm>
00026
00027 template <typename T>
00028 class AAX_CBinaryDisplayDelegate : public AAX_IDisplayDelegate<T>
00029 {
00030 public:
00031     AAX_CBinaryDisplayDelegate(const char* falseString, const char* trueString);
00032     AAX_CBinaryDisplayDelegate(const AAX_CBinaryDisplayDelegate& other);
00033
00034     //Virtual Overrides
00035     AAX_IDisplayDelegate<T>* Clone() const AAX_OVERRIDE;
00036     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00037     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString)
00038     const AAX_OVERRIDE;
00039     bool StringToValue(const AAX_CString& valueString, T* value) const
00040     AAX_OVERRIDE;
00041
00042     // AAX_CBinaryDisplayDelegate
00043     virtual void AddShortenedStrings(const char* falseString, const char*
00044     trueString, int iStrLength);
00045
00046 private:
00047     AAX_CBinaryDisplayDelegate(); //private constructor to prevent its use externally.
00048
00049     const AAX_CString mFalseString;
00050     const AAX_CString mTrueString;
00051     uint32_t mMaxStrLength;
00052
00053     struct StringTable
00054     {
00055         int mStrLength;
00056         AAX_CString mFalseString;
00057         AAX_CString mTrueString;
00058     };
00059     static bool StringTableSortFunc(struct StringTable i, struct StringTable j)
00060     {
00061         return (i.mStrLength < j.mStrLength);
00062     }
00063
00064     std::vector<struct StringTable> mShortenedStrings;
00065 };
00066
00067 template <typename T>
00068 AAX_CBinaryDisplayDelegate<T>::AAX_CBinaryDisplayDelegate(const char* falseString, const char*
00069 trueString) :
00070     mFalseString(falseString),
00071     mTrueString(trueString),
00072     mMaxStrLength(0)
00073 {
00074     mMaxStrLength = (std::max)(mMaxStrLength, mFalseString.Length());
00075     mMaxStrLength = (std::max)(mMaxStrLength, mTrueString.Length());
00076 }
00077
00078 template <typename T>
00079 AAX_CBinaryDisplayDelegate<T>::AAX_CBinaryDisplayDelegate(const AAX_CBinaryDisplayDelegate& other) :
00080     mFalseString(other.mFalseString),
00081     mTrueString(other.mTrueString),
00082     mMaxStrLength(other.mMaxStrLength)
00083 {
00084     if ( other.mShortenedStrings.size() > 0 )
00085     {
00086         for ( size_t i = 0; i < other.mShortenedStrings.size(); i++ )
00087             mShortenedStrings.push_back( other.mShortenedStrings.at(i) );
00088     }
00089 }

```

```

00112     }
00113 }
00114
00115 template <typename T>
00116 void AAX_CBinaryDisplayDelegate<T>::AddShortenedStrings(const char* falseString, const char*
trueString, int iStrLength)
00117 {
00118     struct StringTable shortendTable;
00119     shortendTable.mStrLength = iStrLength;
00120     shortendTable.mFalseString = AAX_CString(falseString);
00121     shortendTable.mTrueString = AAX_CString(trueString);
00122     mShortenedStrings.push_back(shortendTable);
00123
00124     // keep structure sorted by str lengths
00125     std::sort(mShortenedStrings.begin(), mShortenedStrings.end(),
AAX_CBinaryDisplayDelegate::StringTableSortFunc );
00126 }
00127
00128
00129 template <typename T>
00130 AAX_IDisplayDelegate<T>* AAX_CBinaryDisplayDelegate<T>::Clone() const
00131 {
00132     return new AAX_CBinaryDisplayDelegate(*this);
00133 }
00134
00135 template <typename T>
00136 bool AAX_CBinaryDisplayDelegate<T>::ValueToString(T value, AAX_CString* valueString) const
00137 {
00138     if (value)
00139         *valueString = mTrueString;
00140     else
00141         *valueString = mFalseString;
00142     return true;
00143 }
00144
00145 template <typename T>
00146 bool AAX_CBinaryDisplayDelegate<T>::ValueToString(T value, int32_t maxNumChars, AAX_CString*
valueString) const
00147 {
00148     // if we don't have any shortened strings, just return the full length version
00149     if ( mShortenedStrings.size() == 0 )
00150         return this->ValueToString(value, valueString);
00151
00152     // first see if requested length is longer than normal strings
00153     const uint32_t maxNumCharsUnsigned = (0 <= maxNumChars) ? static_cast<uint32_t>(maxNumChars) : 0;
00154     if ( maxNumCharsUnsigned >= mMaxStrLength )
00155     {
00156         if (value)
00157             *valueString = mTrueString;
00158         else
00159             *valueString = mFalseString;
00160         return true;
00161     }
00162
00163     // iterate through shortened strings from longest to shortest
00164     // taking the first set that is short enough
00165     for ( int i = static_cast<int>(mShortenedStrings.size())-1; i >= 0; i-- )
00166     {
00167         struct StringTable shortStrings = mShortenedStrings.at(static_cast<unsigned int>(i));
00168         if ( shortStrings.mStrLength <= maxNumChars )
00169         {
00170             if (value)
00171                 *valueString = shortStrings.mTrueString;
00172             else
00173                 *valueString = shortStrings.mFalseString;
00174             return true;
00175         }
00176     }
00177
00178     // if we can't find one short enough, just use the shortest version we can find
00179     struct StringTable shortestStrings = mShortenedStrings.at(0);
00180     if (value)
00181         *valueString = shortestStrings.mTrueString;
00182     else
00183         *valueString = shortestStrings.mFalseString;
00184     return true;
00185 }
00186 }
00187
00188 template <typename T>
00189 bool AAX_CBinaryDisplayDelegate<T>::StringToValue(const AAX_CString& valueString, T* value) const
00190 {
00191     if (valueString == mTrueString)
00192     {
00193         *value = (T)(true);
00194         return true;
00195     }

```

```

00196     if (valueString == mFalseString)
00197     {
00198         *value = (T)(false);
00199         return true;
00200     }
00201     *value = (T)(false);
00202     return false;
00203 }
00204
00205
00206
00207
00208 #endif //AAX_CBINARYDISPLAYDELEGATE_H

```

15.69 AAX_CBinaryTaperDelegate.h File Reference

```
#include "AAX_ITaperDelegate.h"
```

15.69.1 Description

A binary taper delegate.

Classes

- class [AAX_CBinaryTaperDelegate< T >](#)
A binary taper conforming to [AAX_ITaperDelegate](#).

15.70 AAX_CBinaryTaperDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CBINARYTAPERDELEGATE_H
00023 #define AAX_CBINARYTAPERDELEGATE_H
00024
00025 #include "AAX_ITaperDelegate.h"
00026
00027
00040 template <typename T>
00041 class AAX_CBinaryTaperDelegate : public AAX_ITaperDelegate<T>
00042 {
00043 public:
00044
00048     AAX_CBinaryTaperDelegate( );
00049
00050     //Virtual Overrides
00051     AAX_ITaperDelegate<T>* Clone() const AAX_OVERRIDE;
00052     T GetMaximumValue() const AAX_OVERRIDE;
00053     T GetMinimumValue() const AAX_OVERRIDE;
00054     T ConstrainRealValue(T value) const AAX_OVERRIDE;
00055     T NormalizedToReal(double normalizedValue) const AAX_OVERRIDE;

```

```

00056     double                               RealToNormalized(T realValue) const AAX_OVERRIDE;
00057 };
00058
00059
00060
00061
00062
00063
00064 template <typename T>
00065 AAX_CBinaryTaperDelegate<T>::AAX_CBinaryTaperDelegate( )      :
00066     AAX_ITaperDelegate<T>()
00067 {
00068 }
00069
00070 template <typename T>
00071 AAX_ITaperDelegate<T>* AAX_CBinaryTaperDelegate<T>::Clone() const
00072 {
00073     return new AAX_CBinaryTaperDelegate(*this);
00074 }
00075
00076 template <typename T>
00077 T AAX_CBinaryTaperDelegate<T>::GetMinimumValue() const
00078 {
00079     return false;
00080 }
00081
00082 template <typename T>
00083 T AAX_CBinaryTaperDelegate<T>::GetMaximumValue() const
00084 {
00085     return true;
00086 }
00087
00088 template <typename T>
00089 T AAX_CBinaryTaperDelegate<T>::ConstrainRealValue(T value) const
00090 {
00091     return value;
00092 }
00093
00094 template <typename T>
00095 T AAX_CBinaryTaperDelegate<T>::NormalizedToReal(double normalizedValue) const
00096 {
00097     if (normalizedValue > 0.0f)
00098         return (T)(1);           //should construct true for bool
00099     return (T)(0);               //should construct false for bool
00100 }
00101
00102 template <typename T>
00103 double AAX_CBinaryTaperDelegate<T>::RealToNormalized(T realValue) const
00104 {
00105     if (realValue > (T)(0))
00106         return 1.0f;
00107     return 0.0f;
00108 }
00109
00110
00111
00112
00113 #endif //AAX_CBINARYTAPERDELEGATE_H
00114
00115

```

15.71 AAX_CChunkDataParser.h File Reference

```

#include "AAX.h"
#include "AAX_CString.h"
#include <vector>

```

15.71.1 Description

Parser utility for plugin chunks.

Classes

- class [AAX_CChunkDataParser](#)
Parser utility for plugin chunks.
- struct [AAX_CChunkDataParser::DataValue](#)

Namespaces

- namespace [AAX_ChunkDataParserDefs](#)
Constants used by ChunkDataParser class.

Macros

- `#define` [AAX_CHUNKDATAPARSER_H](#)

Variables

- `const int32_t` [AAX_ChunkDataParserDefs::FLOAT_TYPE](#) = 1
- `const char` [AAX_ChunkDataParserDefs::FLOAT_STRING_IDENTIFIER](#) [] = "f_"
- `const int32_t` [AAX_ChunkDataParserDefs::LONG_TYPE](#) = 2
- `const char` [AAX_ChunkDataParserDefs::LONG_STRING_IDENTIFIER](#) [] = "l_"
- `const int32_t` [AAX_ChunkDataParserDefs::DOUBLE_TYPE](#) = 3
- `const char` [AAX_ChunkDataParserDefs::DOUBLE_STRING_IDENTIFIER](#) [] = "d_"
- `const size_t` [AAX_ChunkDataParserDefs::DOUBLE_TYPE_SIZE](#) = 8
- `const size_t` [AAX_ChunkDataParserDefs::DOUBLE_TYPE_INCR](#) = 8
- `const int32_t` [AAX_ChunkDataParserDefs::SHORT_TYPE](#) = 4
- `const char` [AAX_ChunkDataParserDefs::SHORT_STRING_IDENTIFIER](#) [] = "s_"
- `const size_t` [AAX_ChunkDataParserDefs::SHORT_TYPE_SIZE](#) = 2
- `const size_t` [AAX_ChunkDataParserDefs::SHORT_TYPE_INCR](#) = 4
- `const int32_t` [AAX_ChunkDataParserDefs::STRING_TYPE](#) = 5
- `const char` [AAX_ChunkDataParserDefs::STRING_STRING_IDENTIFIER](#) [] = "r_"
- `const size_t` [AAX_ChunkDataParserDefs::MAX_STRINGDATA_LENGTH](#) = 255
- `const size_t` [AAX_ChunkDataParserDefs::DEFAULT32BIT_TYPE_SIZE](#) = 4
- `const size_t` [AAX_ChunkDataParserDefs::DEFAULT32BIT_TYPE_INCR](#) = 4
- `const size_t` [AAX_ChunkDataParserDefs::STRING_IDENTIFIER_SIZE](#) = 2
- `const int32_t` [AAX_ChunkDataParserDefs::NAME_NOT_FOUND](#) = -1
- `const size_t` [AAX_ChunkDataParserDefs::MAX_NAME_LENGTH](#) = 255
- `const int32_t` [AAX_ChunkDataParserDefs::BUILD_DATA_FAILED](#) = -333
- `const int32_t` [AAX_ChunkDataParserDefs::HEADER_SIZE](#) = 4
- `const int32_t` [AAX_ChunkDataParserDefs::VERSION_ID_1](#) = 0x01010101

15.71.2 Macro Definition Documentation

15.71.2.1 AAX_CHUNKDATAPARSER_H

```
#define AAX_CHUNKDATAPARSER_H
```

15.72 AAX_CChunkDataParser.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00019  /*=====*/
00020
00021
00022  #pragma once
00023
00024  #ifndef AAX_CHUNKDATAPARSER_H
00025  #define AAX_CHUNKDATAPARSER_H
00026
00027  #include "AAX.h"
00028  #include "AAX_CString.h"
00029  #include <vector>
00030
00031  //forward declarations
00032  struct AAX_SPlugInChunk;
00033
00037  namespace AAX_ChunkDataParserDefs {
00038      const int32_t FLOAT_TYPE = 1;
00039      const char FLOAT_STRING_IDENTIFIER[] = "f_";
00040
00041      const int32_t LONG_TYPE = 2;
00042      const char LONG_STRING_IDENTIFIER[] = "l_";
00043
00044      const int32_t DOUBLE_TYPE = 3;
00045      const char DOUBLE_STRING_IDENTIFIER[] = "d_";
00046      const size_t DOUBLE_TYPE_SIZE = 8;
00047      const size_t DOUBLE_TYPE_INCR = 8;
00048
00049      const int32_t SHORT_TYPE = 4;
00050      const char SHORT_STRING_IDENTIFIER[] = "s_";
00051      const size_t SHORT_TYPE_SIZE = 2;
00052      const size_t SHORT_TYPE_INCR = 4; // keep life word aligned
00053
00054      const int32_t STRING_TYPE = 5;
00055      const char STRING_STRING_IDENTIFIER[] = "r_";
00056      const size_t MAX_STRINGDATA_LENGTH = 255;
00057
00058      const size_t DEFAULT32BIT_TYPE_SIZE = 4;
00059      const size_t DEFAULT32BIT_TYPE_INCR = 4;
00060
00061      const size_t STRING_IDENTIFIER_SIZE = 2;
00062
00063      const int32_t NAME_NOT_FOUND = -1;
00064      const size_t MAX_NAME_LENGTH = 255;
00065      const int32_t BUILD_DATA_FAILED = -333;
00066      const int32_t HEADER_SIZE = 4;
00067      const int32_t VERSION_ID_1 = 0x01010101;
00068  }
00069
00114  class AAX_CChunkDataParser
00115  {
00116      public:
00117          AAX_CChunkDataParser();
00118          virtual ~AAX_CChunkDataParser();
00119
00125          void AddFloat(const char *name, float value);
00126          void AddDouble(const char *name, double value);
00127          void AddInt32(const char *name, int32_t value);
00128          void AddInt16(const char *name, int16_t value);
00129          void AddString(const char *name, AAX_CString value);
00130
00136          bool FindFloat(const char *name, float *value);
00137          bool FindDouble(const char *name, double *value);
00138          bool FindInt32(const char *name, int32_t *value);
00139          bool FindInt16(const char *name, int16_t *value);
00140          bool FindString(const char *name, AAX_CString *value);
00141
00142          bool ReplaceDouble(const char *name, double value); //SW added for fela
00143          int32_t GetChunkData(AAX_SPlugInChunk *chunk);
00144          int32_t GetChunkDataSize();
00145          int32_t GetChunkVersion() {return mChunkVersion;}

```

```

00146         bool    IsEmpty();
00147         void    Clear();
00149
00152         void    LoadChunk(const AAX_SPlugInChunk *chunk);
00153
00154     protected:
00155         void    WordAlign(uint32_t &index);
00156         void    WordAlign(int32_t &index);
00157         int32_t FindName(const AAX_CString &Name);
00159
00166         int32_t mLastFoundIndex;
00167
00168         char    *mChunkData;
00169
00170         int32_t mChunkVersion;
00171     public:
00172         struct DataValue
00173         {
00174             int32_t    mDataType;
00175             AAX_CString mDataName;
00176             int64_t    mIntValue;
00177             AAX_CString mStringValue;
00178
00179             DataValue():
00180                 mDataType(0),
00181                 mDataName(AAX_CString()),
00182                 mIntValue(0),
00183                 mStringValue(AAX_CString())
00184             {};
00185         };
00186
00187         std::vector<DataValue> mDataValues;
00188     };
00189
00190 #endif //AAX_CHUNKDATAPARSER_H

```

15.73 AAX_CDecibelDisplayDelegateDecorator.h File Reference

```

#include "AAX_IDisplayDelegateDecorator.h"
#include <cmath>

```

15.73.1 Description

A decibel display delegate.

Classes

- class [AAX_CDecibelDisplayDelegateDecorator< T >](#)
A percent decorator conforming to [AAX_IDisplayDelegateDecorator](#).

15.74 AAX_CDecibelDisplayDelegateDecorator.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *

```



```

00011  */
00012
00019  /*=====*/
00020
00021
00022 #ifndef AAX_CDECIBELDISPLAYDELEGATEDECORATOR_H
00023 #define AAX_CDECIBELDISPLAYDELEGATEDECORATOR_H
00024
00025
00026 #include "AAX_IDisplayDelegateDecorator.h"
00027 #include <cmath>
00028
00029
00030
00052 template <typename T>
00053 class AAX_CDecibelDisplayDelegateDecorator : public AAX_IDisplayDelegateDecorator<T>
00054 {
00055 public:
00056     AAX_CDecibelDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>& displayDelegate);
00057
00058     //Virtual Overrides
00059     AAX_CDecibelDisplayDelegateDecorator<T>* Clone() const AAX_OVERRIDE;
00060     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00061     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const
00062     AAX_OVERRIDE;
00063     bool StringToValue(const AAX_CString& valueString, T* value) const AAX_OVERRIDE;
00064 };
00065
00066
00067
00068
00069
00070 template <typename T>
00071 AAX_CDecibelDisplayDelegateDecorator<T>::AAX_CDecibelDisplayDelegateDecorator(const
00072     AAX_IDisplayDelegate<T>& displayDelegate) :
00073     AAX_IDisplayDelegateDecorator<T>(displayDelegate)
00074 {
00075 }
00076
00077 template <typename T>
00078 AAX_CDecibelDisplayDelegateDecorator<T>* AAX_CDecibelDisplayDelegateDecorator<T>::Clone() const
00079 {
00080     return new AAX_CDecibelDisplayDelegateDecorator(*this);
00081 }
00082
00083 template <typename T>
00084 bool AAX_CDecibelDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString)
00085 const
00086 {
00087     bool succeeded = false;
00088     if (value <= 0)
00089     {
00090         //valueString = AAX_CString("--- dB");
00091         *valueString = AAX_CString("-INF ");
00092         succeeded = true;
00093     }
00094     else
00095     {
00096         value = (T)(20.0*log10(value));
00097         if ( value > -0.01f && value < 0.0f) //To prevent minus for 0.0 value in automation turned on
00098             value = 0.0f;
00099         succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00100     }
00101     *valueString += AAX_CString("dB");
00102     return succeeded;
00103 }
00104
00105
00106 template <typename T>
00107 bool AAX_CDecibelDisplayDelegateDecorator<T>::ValueToString(T value, int32_t maxNumChars,
00108     AAX_CString* valueString) const
00109 {
00110     if (value <= 0)
00111     {
00112         *valueString = AAX_CString("-INF");
00113         if (maxNumChars >= 7)
00114             valueString->Append(" dB"); //<DMT> Add a space for longer strings and dB
00115         return true;
00116     }
00117     value = (T)(20.0*log10(value));
00118     if ( value > -0.01f && value < 0.0f) //To prevent minus for 0.0 value in automation turned on
00119         value = 0.0f;
00120     bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars, valueString);

```

```

00121
00122
00123     //<DMT> Check current string length and see if there is room to add units.  I believe these units
are usually less important than precision on control surfaces.
00124     uint32_t strlen = valueString->Length();
00125     const uint32_t maxNumCharsUnsigned = (0 <= maxNumChars) ? static_cast<uint32_t>(maxNumChars) : 0;
00126     if (maxNumCharsUnsigned >= (strlen + 2))    //length of string plus 2 for the "dB"
00127         *valueString += AAX_CString("dB");
00128     return succeeded;
00129 }
00130
00131
00132 template <typename T>
00133 bool    AAX_CDecibelDisplayDelegateDecorator<T>::StringToValue(const AAX_CString& valueString, T*
value) const
00134 {
00135     //Just call through if there is obviously no unit string.
00136     if (valueString.Length() <= 2)
00137     {
00138         bool success = AAX_IDisplayDelegateDecorator<T>::StringToValue(valueString, value);
00139         *value = (T)pow((T)10.0, (*value / (T)20.0));
00140         return success;
00141     }
00142
00143     //Just call through if the end of this string does not match the unit string.
00144     AAX_CString unitSubString;
00145     valueString.SubString(valueString.Length() - 2, 2, &unitSubString);
00146     if (unitSubString != AAX_CString("dB"))
00147     {
00148         bool success = AAX_IDisplayDelegateDecorator<T>::StringToValue(valueString, value);
00149         *value = (T)pow((T)10.0, *value / (T)20.0);
00150         return success;
00151     }
00152
00153     //Call through with the stripped down value string.
00154     AAX_CString valueSubString;
00155     valueString.SubString(0, valueString.Length() - 2, &valueSubString);
00156     bool success = AAX_IDisplayDelegateDecorator<T>::StringToValue(valueSubString, value);
00157     *value = (T)pow((T)10.0, *value / (T)20.0);
00158     return success;
00159 }
00160
00161
00162
00163
00164 #endif //AAX_CDECIBELDISPLAYDELEGATEDECORATOR_H

```

15.75 AAX_CEffectDirectData.h File Reference

```
#include "AAX_IEffectDirectData.h"
```

15.75.1 Description

A default implementation of the [AAX_IEffectDirectData](#) interface.

Classes

- class [AAX_CEffectDirectData](#)
Default implementation of the [AAX_IEffectDirectData](#) interface.

Macros

- #define [AAX_CEFFECTDIRECTDATA_H](#)

15.75.2 Macro Definition Documentation

15.75.2.1 AAX_CEFFECTDIRECTDATA_H

```
#define AAX_CEFFECTDIRECTDATA_H
```

15.76 AAX_CEffectDirectData.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020
00021 #pragma once
00022 #ifndef AAX_CEFFECTDIRECTDATA_H
00023 #define AAX_CEFFECTDIRECTDATA_H
00024
00025 #include "AAX_IEffectDirectData.h"
00026
00027
00028
00029 class AAX_IPrivateDataAccess;
00030 class AAX_IEffectParameters;
00031 class AAX_IController;
00032
00033
00034
00042 class AAX_CEffectDirectData : public AAX_IEffectDirectData
00043 {
00044 public:
00045
00046     AAX_CEffectDirectData(
00047         void);
00048
00049     virtual
00050     ~AAX_CEffectDirectData(
00051         void);
00052
00053 public:
00054
00069     AAX_Result Initialize (IACFUnknown * iController ) AAX_OVERRIDE AAX_FINAL;
00070     AAX_Result Uninitialize (void) AAX_OVERRIDE;
00072
00088     AAX_Result TimerWakeup (IACFUnknown * iDataAccessInterface ) AAX_OVERRIDE;
00090
00115     AAX_Result NotificationReceived( AAX_CTypeID inNotificationType,
00116                                     const void * inNotificationData,
00117                                     uint32_t inNotificationDataSize) AAX_OVERRIDE;
00119
00120
00121 public:
00122
00131     AAX_IController* Controller (void);
00137     AAX_IEffectParameters* EffectParameters (void);
00139
00140 protected:
00141
00152     virtual AAX_Result Initialize_PrivateDataAccess();
00161     virtual AAX_Result TimerWakeup_PrivateDataAccess (AAX_IPrivateDataAccess* iPrivateDataAccess);
00163
00164 private:
00165     AAX_IController* mController;
00166     AAX_IEffectParameters* mEffectParameters;
00167 };
00168
00169
00170 #endif // AAX_CEFFECTDIRECTDATA_H
```

15.77 AAX_CEffectGUI.h File Reference

```
#include "AAX_IEffectGUI.h"
#include "AAX_IACFEffectParameters.h"
#include <string>
#include <vector>
#include <map>
#include <memory>
```

15.77.1 Description

A default implementation of the [AAX_IEffectGUI](#) interface.

Classes

- class [AAX_CEffectGUI](#)

Default implementation of the [AAX_IEffectGUI](#) interface.

15.78 AAX_CEffectGUI.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CEFFECTGUI_H
00023 #define AAX_CEFFECTGUI_H
00024
00025 #include "AAX_IEffectGUI.h"
00026 #include "AAX_IACFEffectParameters.h"
00027
00028 #include <string>
00029 #include <vector>
00030 #include <map>
00031 #include <memory>
00032
00033
00034 class AAX_IEffectParameters;
00035 class AAX_IController;
00036 class AAX_IViewContainer;
00037 class AAX_ITransport;
00038
00039
00040
00054 class AAX_CEffectGUI : public AAX_IEffectGUI
00055 {
00056 public:
00057
00058     AAX_CEffectGUI(void);
00059     ~AAX_CEffectGUI(void) AAX_OVERRIDE;
00060
00061 public:
00062
00066     AAX_Result Initialize (IACFUnknown * iController ) AAX_OVERRIDE;
```

```

00067     AAX_Result      Uninitialize (void) AAX_OVERRIDE;
00069
00079     AAX_Result      NotificationReceived(AAX_CTypeID inNotificationType, const void *
inNotificationData, uint32_t inNotificationDataSize) AAX_OVERRIDE;
00081
00085     AAX_Result SetViewContainer (IACFUnknown * iViewContainer ) AAX_OVERRIDE;
00086     AAX_Result GetViewSize (AAX_Point * /* oViewSize */ ) const AAX_OVERRIDE
00087     {
00088         return AAX_SUCCESS;
00089     }
00091
00095     AAX_Result Draw (AAX_Rect * /* iDrawRect */ ) AAX_OVERRIDE
00096     {
00097         return AAX_SUCCESS;
00098     }
00099     AAX_Result TimerWakeUp (void) AAX_OVERRIDE
00100     {
00101         return AAX_SUCCESS;
00102     }
00103     AAX_Result ParameterUpdated(AAX_CParamID paramID) AAX_OVERRIDE;
00105
00111     AAX_Result      GetCustomLabel ( AAX_EPlugInStrings iSelector, AAX_IString * oString ) const
AAX_OVERRIDE;
00112
00113     AAX_Result SetControlHighlightInfo (AAX_CParamID /* iParameterID */, AAX_CBoolean /*
iIsHighlighted */, AAX_EHighlightColor /* iColor */) AAX_OVERRIDE
00114     {
00115         return AAX_SUCCESS;
00116     }
00118
00119 protected:
00120
00134     virtual void CreateViewContents (void) = 0;
00141     virtual void CreateViewContainer (void) = 0;
00149     virtual void DeleteViewContainer (void) = 0;
00151
00166     virtual void UpdateAllParameters (void);
00168
00169 public: //These accessors are public here as they are often needed by contained views.
00170
00178     AAX_IController* GetController (void);
00179     const AAX_IController* GetController (void) const;
00180
00185     AAX_IEffectParameters* GetEffectParameters (void);
00186     const AAX_IEffectParameters* GetEffectParameters (void) const;
00187
00192     AAX_IViewContainer* GetViewContainer (void);
00193     const AAX_IViewContainer* GetViewContainer (void) const;
00194
00199     AAX_ITransport*      Transport ();
00200     const AAX_ITransport*      Transport () const;
00201
00206     AAX_EViewContainer_Type      GetViewContainerType ();
00207     void *      GetViewContainerPtr ();
00209
00210 private:
00211     //These are private, but they all have protected accessors.
00212     AAX_IController *      mController;
00213     AAX_IEffectParameters *      mEffectParameters;
00214     AAX_UNIQUE_PTR(AAX_IViewContainer) mViewContainer;
00215     AAX_ITransport*      mTransport;
00216 };
00217
00218
00219 #endif

```

15.79 AAX_CEffectParameters.h File Reference

```

#include "AAX_IEffectParameters.h"
#include "AAX_IPageTable.h"
#include "AAX_CString.h"
#include "AAX_CChunkDataParser.h"
#include "AAX_CParameterManager.h"
#include "AAX_CPacketDispatcher.h"
#include <set>
#include <string>
#include <vector>

```

15.79.1 Description

A default implementation of the `AAX_IeffectParameters` interface.

Classes

- class [AAX_CEffectParameters](#)
Default implementation of the [AAX_IEffectParameters](#) interface.

Functions

- `int32_t` [NormalizedToInt32](#) (`double` `normalizedValue`)
- `double` [Int32ToNormalized](#) (`int32_t` `value`)
- `double` [BoolToNormalized](#) (`bool` `value`)

Variables

- [AAX_CParamID](#) `cPreviewID`
- [AAX_CParamID](#) `cDefaultMasterBypassID`

15.79.2 Function Documentation

15.79.2.1 NormalizedToInt32()

```
int32_t NormalizedToInt32 (  
    double normalizedValue )
```

15.79.2.2 Int32ToNormalized()

```
double Int32ToNormalized (  
    int32_t value )
```

15.79.2.3 BoolToNormalized()

```
double BoolToNormalized (  
    bool value )
```

15.79.3 Variable Documentation

15.79.3.1 cPreviewID

AAX_CParamID cPreviewID

15.79.3.2 cDefaultMasterBypassID

AAX_CParamID cDefaultMasterBypassID

15.80 AAX_CEffectParameters.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00019  /*=====*/
00020
00021
00022  #ifndef AAX_CEFFECTPARAMETERS_H
00023  #define AAX_CEFFECTPARAMETERS_H
00024
00025  #include "AAX_IEffectParameters.h"
00026  #include "AAX_IPageTable.h"
00027  #include "AAX_CString.h"
00028  #include "AAX_CChunkDataParser.h"
00029  #include "AAX_CParameterManager.h"
00030  #include "AAX_CPacketDispatcher.h"
00031
00032  #include <set>
00033  #include <string>
00034  #include <vector>
00035
00036  class AAX_IController;
00037  class AAX_IAutomationDelegate;
00038  class AAX_CParameterManager;
00039  class AAX_CPacketDispatcher;
00040  class AAX_ITransport;
00041
00042  extern "C" AAX_CParamID cPreviewID;
00043  extern "C" AAX_CParamID cDefaultMasterBypassID;
00044
00062  class AAX_CEffectParameters : public AAX_IEffectParameters
00063  {
00064  public:
00065      AAX_CEffectParameters (void);
00066      ~AAX_CEffectParameters (void) AAX_OVERRIDE;
00067      AAX_CEffectParameters& operator= (const AAX_CEffectParameters& other);
00068
00069  public:
00079      AAX_Result Initialize(IACFUnknown* iController) AAX_OVERRIDE;
00080      AAX_Result Uninitialize (void) AAX_OVERRIDE;
00082
00086      AAX_Result NotificationReceived( /* AAX_ENotificationEvent */ AAX_CTypeID inNotificationType,
00087                                     const void * inNotificationData, uint32_t inNotificationDataSize) AAX_OVERRIDE;
00088

```

```

00097     AAX_Result GetNumberOfParameters (int32_t * oNumControls) const AAX_OVERRIDE;
00098     AAX_Result GetMasterBypassParameter (AAX_IString * oIDString) const AAX_OVERRIDE;
00099     AAX_Result GetParameterIsAutomatable (AAX_CParamID iParameterID, AAX_CBoolean * oAutomatable)
const AAX_OVERRIDE;
00100     AAX_Result GetParameterNumberOfSteps (AAX_CParamID iParameterID, int32_t * oNumSteps ) const
AAX_OVERRIDE;
00101     AAX_Result GetParameterName (AAX_CParamID iParameterID, AAX_IString * oName ) const AAX_OVERRIDE;
00102     AAX_Result GetParameterNameOfLength (AAX_CParamID iParameterID, AAX_IString * oName, int32_t
iNameLength ) const AAX_OVERRIDE;
00103     AAX_Result GetParameterDefaultNormalizedValue (AAX_CParamID iParameterID, double * oValue ) const
AAX_OVERRIDE;
00104     AAX_Result SetParameterDefaultNormalizedValue (AAX_CParamID iParameterID, double iValue )
AAX_OVERRIDE;
00105     AAX_Result GetParameterType (AAX_CParamID iParameterID, AAX_EParameterType * oParameterType )
const AAX_OVERRIDE;
00106     AAX_Result GetParameterOrientation (AAX_CParamID iParameterID, AAX_EParameterOrientation *
oParameterOrientation ) const AAX_OVERRIDE;
00107     AAX_Result GetParameter (AAX_CParamID iParameterID, AAX_IParameter ** oParameter ) AAX_OVERRIDE;
00108     AAX_Result GetParameterIndex (AAX_CParamID iParameterID, int32_t * oControlIndex ) const
AAX_OVERRIDE;
00109     AAX_Result GetParameterIDFromIndex (int32_t iControlIndex, AAX_IString * oParameterIDString )
const AAX_OVERRIDE;
00110     AAX_Result GetParameterTypeInfo ( AAX_CParamID iParameterID, int32_t iSelector, int32_t* oValue)
const AAX_OVERRIDE;
00112
00121     AAX_Result GetParameterValueFromString (AAX_CParamID iParameterID, double * oValue, const
AAX_IString & iValueString ) const AAX_OVERRIDE;
00122     AAX_Result GetParameterStringFromValue (AAX_CParamID iParameterID, double iValue, AAX_IString *
oValueString, int32_t iMaxLength ) const AAX_OVERRIDE;
00123     AAX_Result GetParameterValueString (AAX_CParamID iParameterID, AAX_IString * oValueString, int32_t
iMaxLength) const AAX_OVERRIDE;
00124     AAX_Result GetParameterNormalizedValue (AAX_CParamID iParameterID, double * oValuePtr ) const
AAX_OVERRIDE;
00125     AAX_Result SetParameterNormalizedValue (AAX_CParamID iParameterID, double iValue ) AAX_OVERRIDE;
00126     AAX_Result SetParameterNormalizedRelative (AAX_CParamID iParameterID, double iValue )
AAX_OVERRIDE;
00128
00139     AAX_Result TouchParameter ( AAX_CParamID iParameterID ) AAX_OVERRIDE;
00140     AAX_Result ReleaseParameter ( AAX_CParamID iParameterID ) AAX_OVERRIDE;
00141     AAX_Result UpdateParameterTouch ( AAX_CParamID iParameterID, AAX_CBoolean iTouchState )
AAX_OVERRIDE;
00143
00161     AAX_Result UpdateParameterNormalizedValue (AAX_CParamID iParameterID, double iValue,
AAX_EUpdateSource iSource ) AAX_OVERRIDE;
00162     AAX_Result UpdateParameterNormalizedRelative (AAX_CParamID iParameterID, double iValue )
AAX_OVERRIDE;
00163     AAX_Result GenerateCoefficients(void) AAX_OVERRIDE;
00165
00169     AAX_Result ResetFieldData (AAX_CFieldIndex inFieldIndex, void * oData, uint32_t inDataSize) const
AAX_OVERRIDE;
00171
00190     AAX_Result GetNumberOfChunks (int32_t * oNumChunks ) const AAX_OVERRIDE;
00191     AAX_Result GetChunkIDFromIndex (int32_t iIndex, AAX_CTypeID * oChunkID ) const AAX_OVERRIDE;
00192     AAX_Result GetChunkSize (AAX_CTypeID iChunkID, uint32_t * oSize ) const AAX_OVERRIDE;
00193     AAX_Result GetChunk (AAX_CTypeID iChunkID, AAX_SPlugInChunk * oChunk ) const AAX_OVERRIDE;
00194     AAX_Result SetChunk (AAX_CTypeID iChunkID, const AAX_SPlugInChunk * iChunk ) AAX_OVERRIDE;
00195     AAX_Result CompareActiveChunk (const AAX_SPlugInChunk * iChunkP, AAX_CBoolean * oIsEqual ) const
AAX_OVERRIDE;
00196     AAX_Result GetNumberOfChanges (int32_t * oNumChanges ) const AAX_OVERRIDE;
00198
00203     AAX_Result TimerWakeup() AAX_OVERRIDE;
00205
00210     AAX_Result GetCurveData( /* AAX_ECurveType */ AAX_CTypeID iCurveType, const float * iValues,
uint32_t iNumValues, float * oValues ) const AAX_OVERRIDE;
00211     AAX_Result GetCurveDataMeterIds( /* AAX_ECurveType */ AAX_CTypeID iCurveType, uint32_t *oXMeterId,
uint32_t *oYMeterId) const AAX_OVERRIDE;
00212     AAX_Result GetCurveDataDisplayRange( /* AAX_ECurveType */ AAX_CTypeID iCurveType, float *oXMin,
float *oXMax, float *oYMin, float *oYMax ) const AAX_OVERRIDE;
00213
00220     AAX_Result UpdatePageTable(uint32_t inTableType, int32_t inTablePageSize, IACFUnknown*
iHostUnknown, IACFUnknown* ioPageTableUnknown) const AAX_OVERRIDE AAX_FINAL;
00222
00233     AAX_Result GetCustomData( AAX_CTypeID iDataBlockID, uint32_t inDataSize, void* oData,
uint32_t* oDataWritten) const AAX_OVERRIDE;
00234     AAX_Result SetCustomData( AAX_CTypeID iDataBlockID, uint32_t inDataSize, const void*
iData ) AAX_OVERRIDE;
00236
00241     AAX_Result DoMIDITransfers() AAX_OVERRIDE { return AAX_SUCCESS; }
00242     AAX_Result UpdateMIDINodes ( AAX_CFieldIndex inFieldIndex, AAX_CMidiPacket& iPacket )
AAX_OVERRIDE;
00243     AAX_Result UpdateControlMIDINodes ( AAX_CTypeID nodeID, AAX_CMidiPacket& iPacket )
AAX_OVERRIDE;
00245
00250     AAX_Result RenderAudio_Hybrid(AAX_SHybridRenderInfo* ioRenderInfo) AAX_OVERRIDE;
00252
00253
00254

```



```

00255 public:
00260     AAX_IController*          Controller();
00261     const AAX_IController*    Controller() const;
00262     AAX_ITransport*           Transport();
00263     const AAX_ITransport*     Transport() const;
00264     AAX_IAutomationDelegate*  AutomationDelegate();
00265     const AAX_IAutomationDelegate* AutomationDelegate() const;
00267
00268 protected:
00273     AAX_Result                SetTaperDelegate ( AAX_CParamID iParameterID, AAX_ITaperDelegateBase &
                                iTaperDelegate, bool iPreserveValue );
00274     AAX_Result                SetDisplayDelegate ( AAX_CParamID iParameterID, AAX_IDisplayDelegateBase &
                                iDisplayDelegate );
00275     bool                      IsParameterTouched ( AAX_CParamID iParameterID ) const;
00276     bool                      IsParameterLinkReady ( AAX_CParamID inParameterID, AAX_EUpdateSource inSource
                                ) const;
00278
00300     virtual AAX_Result       EffectInit(void) { return AAX_SUCCESS; };
00301
00314     virtual AAX_Result UpdatePageTable(uint32_t /*inTableType*/, int32_t /*inTablePageSize*/,
                                AAX_IPageTable& /*ioPageTable*/) const { return AAX_ERROR_UNIMPLEMENTED; };
00315
00326     void FilterParameterIDOnSave(AAX_CParamID controlId);
00328
00332     void BuildChunkData (void) const;
00333
00334 protected:
00335     int32_t                      mNumPlugInChanges;
00336     mutable int32_t              mChunkSize; //this old behavior isn't const friendly
00337     yet. Consider this a temp variable.
00338     mutable AAX_CChunkDataParser mChunkParser; //this old behavior isn't const friendly
00339     yet. Consider this a temp variable.
00338     int32_t                      mNumChunkedParameters;
00339     AAX_CPacketDispatcher        mPacketDispatcher;
00340     AAX_CParameterManager        mParameterManager;
00341     std::set<std::string>        mFilteredParameters;
00342
00343 private:
00344     // interfaces provided by the host via the IACFUnknown passed to Initialize()
00345     AAX_IController*            mController;
00346     AAX_ITransport*             mTransport;
00347     AAX_IAutomationDelegate*    mAutomationDelegate;
00348
00349 };
00350
00351 // Convenience functions since many legacy plug-ins had internal int32 value representations.
00352 extern int32_t NormalizedToInt32 (double normalizedValue );
00353 extern double Int32ToNormalized (int32_t value );
00354 extern double BoolToNormalized (bool value );
00355
00356 #endif

```

15.81 AAX_CHostProcessor.h File Reference

```

#include "AAX_IEffectParameters.h"
#include "AAX_IHostProcessor.h"
#include "ACFPtr.h"

```

15.81.1 Description

Concrete implementation of the [AAX_IHostProcessor](#) interface for non-real-time processing.

Classes

- class [AAX_CHostProcessor](#)

Concrete implementation of the [AAX_IHostProcessor](#) interface for non-real-time processing.

15.82 AAX_CHostProcessor.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00019  /*=====*/
00020
00021
00022  #ifndef AAX_CHOSTPROCESSOR_H
00023  #define AAX_CHOSTPROCESSOR_H
00024
00025  #include "AAX_IEffectParameters.h"
00026  #include "AAX_IHostProcessor.h"
00027  #include "ACFPtr.h"
00028
00029
00030  class AAX_IHostProcessorDelegate;
00031  class AAX_IController;
00032  class AAX_IEffectParameters;
00033  class IACFUnknown;
00034
00054  class AAX_CHostProcessor : public AAX_IHostProcessor
00055  {
00056  public:
00057      /* default constructor */ AAX_CHostProcessor (void);
00058      virtual /* destructor */ ~AAX_CHostProcessor ();
00059
00069      AAX_Result Initialize(IACFUnknown* iController) AAX_OVERRIDE;
00072      AAX_Result Uninitialize() AAX_OVERRIDE;
00074
00117      AAX_Result InitOutputBounds ( int64_t iSrcStart, int64_t iSrcEnd, int64_t * oDstStart,
int64_t * oDstEnd ) AAX_OVERRIDE;
00118
00132      AAX_Result SetLocation ( int64_t iSample ) AAX_OVERRIDE;
00133
00153      AAX_Result RenderAudio ( const float * const inAudioIns [], int32_t inAudioInCount, float *
const iAudioOuts [], int32_t iAudioOutCount, int32_t * ioWindowSize ) AAX_OVERRIDE;
00154
00170      AAX_Result PreRender ( int32_t inAudioInCount, int32_t iAudioOutCount, int32_t iWindowSize
) AAX_OVERRIDE;
00171
00179      AAX_Result PostRender () AAX_OVERRIDE;
00180
00198      AAX_Result AnalyzeAudio ( const float * const inAudioIns [], int32_t inAudioInCount,
int32_t * ioWindowSize ) AAX_OVERRIDE;
00199
00212      AAX_Result PreAnalyze ( int32_t inAudioInCount, int32_t iWindowSize ) AAX_OVERRIDE;
00213
00223      AAX_Result PostAnalyze () AAX_OVERRIDE;
00236      AAX_Result GetClipNameSuffix ( int32_t inMaxLength, AAX_IString* outString ) const
AAX_OVERRIDE;
00238
00239
00243      AAX_IEffectParameters * GetEffectParameters () { return mEffectParameters; }
00244      const AAX_IEffectParameters * GetEffectParameters () const { return mEffectParameters; }
00245      AAX_IHostProcessorDelegate* GetHostProcessorDelegate () { return mHostProcessingDelegate; }
00246      const AAX_IHostProcessorDelegate* GetHostProcessorDelegate () const { return
mHostProcessingDelegate; }
00247
00256      int64_t GetLocation() const { return mLocation; }
00257
00260      int64_t GetInputRange() const { return (mSrcEnd - mSrcStart); }
00263      int64_t GetOutputRange() const { return (mDstEnd - mDstStart); }
00267      int64_t GetSrcStart() const { return mSrcStart; }
00271      int64_t GetSrcEnd() const { return mSrcEnd; }
00278      int64_t GetDstStart() const { return mDstStart; }
00285      int64_t GetDstEnd() const { return mDstEnd; }
00287
00288  protected:
00311      virtual AAX_Result TranslateOutputBounds ( int64_t iSrcStart, int64_t iSrcEnd, int64_t&
oDstStart, int64_t& oDstEnd );
00312
00330      virtual AAX_Result GetAudio ( const float * const inAudioIns [], int32_t inAudioInCount,
int64_t inLocation, int32_t * ioNumSamples );
00331

```

```

00336     virtual int32_t          GetSideChainInputNum ();
00337
00338     // Exterior Object Access
00339     AAX_IController*         Controller()                { return mController; }
00340     const AAX_IController*   Controller() const          { return mController; }
00341     AAX_IHostProcessorDelegate* HostProcessorDelegate() { return
mHostProcessingDelegate; }
00342     const AAX_IHostProcessorDelegate* HostProcessorDelegate() const { return mHostProcessingDelegate;
}
00343     AAX_IEffectParameters*   EffectParameters()          { return mEffectParameters; }
00344     const AAX_IEffectParameters* EffectParameters() const { return mEffectParameters; }
00346
00347 private:
00348     AAX_IController*         mController;
00349     AAX_IHostProcessorDelegate* mHostProcessingDelegate;
00350     AAX_IEffectParameters*   mEffectParameters;
00351     int64_t                  mSrcStart;
00352     int64_t                  mSrcEnd;
00353     int64_t                  mDstStart;
00354     int64_t                  mDstEnd;
00355     int64_t                  mLocation;
00356
00357 };
00358
00359
00360 #endif

```

15.83 AAX_CHostServices.h File Reference

```

#include "AAX.h"
#include "AAX_Enums.h"

```

15.83.1 Description

Concrete implementation of the [AAX_IHostServices](#) interface.

Classes

- class [AAX_CHostServices](#)

Method access to a singleton implementation of the [AAX_IHostServices](#) interface.

15.84 AAX_CHostServices.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2014-2015, 2018, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00019  /*=====*/
00020
00021
00022  #ifndef AAX_CHOSTSERVICES_H
00023  #define AAX_CHOSTSERVICES_H
00024
00025  #include "AAX.h"

```

```

00026 #include "AAX_Enums.h"
00027
00028
00029 class IACFUnknown;
00030
00033 class AAX_CHostServices
00034 {
00035 public:
00036     static void Set ( IACFUnknown * pUnkHost );
00037
00038     static AAX_Result HandleAssertFailure ( const char * iFile, int32_t iLine, const char * iNote, /*
AAX_EAssertFlags */ int32_t iFlags = AAX_eAssertFlags_Default );
00039     static AAX_Result Trace ( AAX_ETracePriorityHost iPriority, const char * iMessage, ... );
00040     static AAX_Result StackTrace ( AAX_ETracePriorityHost iTracePriority, AAX_ETracePriorityHost
iStackTracePriority, const char * iMessage, ... );
00041 };
00042
00043
00044 #endif

```

15.85 AAX_CLinearTaperDelegate.h File Reference

```

#include "AAX_ITaperDelegate.h"
#include "AAX.h"
#include <cmath>

```

15.85.1 Description

A linear taper delegate.

Classes

- class [AAX_CLinearTaperDelegate< T, RealPrecision >](#)
A linear taper conforming to [AAX_ITaperDelegate](#).

15.86 AAX_CLinearTaperDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CLINEARTAPERDELEGATE_H
00023 #define AAX_CLINEARTAPERDELEGATE_H
00024
00025 #include "AAX_ITaperDelegate.h"
00026 #include "AAX.h" //for types
00027
00028 #include <cmath> //for floor()
00029
00030
00056 template <typename T, int32_t RealPrecision=0>

```

```

00057 class AAX_CLinearTaperDelegate : public AAX_ITaperDelegate<T>
00058 {
00059 public:
00060     AAX_CLinearTaperDelegate(T minValue=0, T maxValue=1);
00061
00062     //Virtual AAX_ITaperDelegate Overrides
00063     AAX_CLinearTaperDelegate<T, RealPrecision>* Clone() const AAX_OVERRIDE;
00064     T GetMinimumValue() const AAX_OVERRIDE { return mMinValue; }
00065     T GetMaximumValue() const AAX_OVERRIDE { return mMaxValue; }
00066     T ConstrainRealValue(T value) const AAX_OVERRIDE;
00067     T NormalizedToReal(double normalizedValue) const AAX_OVERRIDE;
00068     double RealToNormalized(T realValue) const AAX_OVERRIDE;
00069
00070 protected:
00071     T Round(double iValue) const;
00072
00073 private:
00074     T mMinValue;
00075     T mMaxValue;
00076 };
00077
00078 template <typename T, int32_t RealPrecision>
00079 T AAX_CLinearTaperDelegate<T, RealPrecision>::Round(double iValue) const
00080 {
00081     double precision = RealPrecision;
00082     if (precision > 0)
00083         return static_cast<T>(floor(iValue * precision + 0.5) / precision);
00084     return static_cast<T>(iValue);
00085 }
00086
00087 template <typename T, int32_t RealPrecision>
00088 AAX_CLinearTaperDelegate<T, RealPrecision>::AAX_CLinearTaperDelegate(T minValue, T maxValue) :
00089     AAX_ITaperDelegate<T>(),
00090     mMinValue(minValue),
00091     mMaxValue(maxValue)
00092 {
00093 }
00094
00095 template <typename T, int32_t RealPrecision>
00096 AAX_CLinearTaperDelegate<T, RealPrecision>* AAX_CLinearTaperDelegate<T, RealPrecision>::Clone()
00097 const
00098 {
00099     return new AAX_CLinearTaperDelegate(*this);
00100 }
00101
00102 template <typename T, int32_t RealPrecision>
00103 T AAX_CLinearTaperDelegate<T, RealPrecision>::ConstrainRealValue(T value) const
00104 {
00105     if (mMinValue == mMaxValue)
00106         return mMinValue;
00107     if (RealPrecision)
00108         value = Round(value); //reduce the precision to get proper rounding behavior with
00109         integers.
00110     const T& highValue = mMaxValue > mMinValue ? mMaxValue : mMinValue;
00111     const T& lowValue = mMaxValue > mMinValue ? mMinValue : mMaxValue;
00112     if (value > highValue)
00113         return highValue;
00114     if (value < lowValue)
00115         return lowValue;
00116     return value;
00117 }
00118
00119 template <typename T, int32_t RealPrecision>
00120 T AAX_CLinearTaperDelegate<T, RealPrecision>::NormalizedToReal(double normalizedValue) const
00121 {
00122     double doubleRealValue = normalizedValue * (double(mMaxValue) - double(mMinValue)) +
00123     double(mMinValue);
00124     // If RealPrecision is set, reduce the precision to get proper rounding behavior with integers.
00125     T realValue = (0 != RealPrecision) ? Round(doubleRealValue) : static_cast<T>(doubleRealValue);
00126     return ConstrainRealValue(realValue);
00127 }
00128
00129 template <typename T, int32_t RealPrecision>
00130 double AAX_CLinearTaperDelegate<T, RealPrecision>::RealToNormalized(T realValue) const
00131 {
00132     realValue = ConstrainRealValue(realValue);
00133     double normalizedValue = (mMaxValue == mMinValue) ? 0.5 : (double(realValue) - double(mMinValue))
00134     / (double(mMaxValue) - double(mMinValue));
00135     return normalizedValue;
00136 }

```

```

00146
00147
00148
00149
00150 #endif //AAX_CLINEARTAPERDELEGATE_H

```

15.87 AAX_CLogTaperDelegate.h File Reference

```

#include "AAX_ITaperDelegate.h"
#include "AAX_UtilsNative.h"
#include "AAX.h"
#include <cmath>

```

15.87.1 Description

A log taper delegate.

Classes

- class [AAX_CLogTaperDelegate< T, RealPrecision >](#)
A logarithmic taper conforming to [AAX_ITaperDelegate](#).

15.88 AAX_CLogTaperDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00018 /*=====*/
00019
00020
00021 #ifndef AAX_CLOGTAPERDELEGATE_H
00022 #define AAX_CLOGTAPERDELEGATE_H
00023
00024 #include "AAX_ITaperDelegate.h"
00025 #include "AAX_UtilsNative.h"
00026 #include "AAX.h" //for types
00027
00028 #include <cmath> //for floor(), log()
00029
00030
00031
00057 template <typename T, int32_t RealPrecision=1000>
00058 class AAX_CLogTaperDelegate : public AAX_ITaperDelegate<T>
00059 {
00060 public:
00068     AAX_CLogTaperDelegate(T minValue=0, T maxValue=1);
00069
00070     //Virtual Overrides
00071     AAX_CLogTaperDelegate<T, RealPrecision>* Clone() const AAX_OVERRIDE;
00072     T GetMinimumValue() const AAX_OVERRIDE { return mMinValue; }
00073     T GetMaximumValue() const AAX_OVERRIDE { return mMaxValue; }
00074     T ConstrainRealValue(T value) const AAX_OVERRIDE;

```

```

00075     T        NormalizedToReal(double normalizedValue) const AAX_OVERRIDE;
00076     double   RealToNormalized(T realValue) const AAX_OVERRIDE;
00077
00078 protected:
00079     T        Round(double iValue) const;
00080
00081 private:
00082     T        mMinValue;
00083     T        mMaxValue;
00084 };
00085
00086 template <typename T, int32_t RealPrecision>
00087 T        AAX_CLogTaperDelegate<T, RealPrecision>::Round(double iValue) const
00088 {
00089     double precision = RealPrecision;
00090     if (precision > 0)
00091         return static_cast<T>(floor(iValue * precision + 0.5) / precision);
00092     return static_cast<T>(iValue);
00093 }
00094
00095 template <typename T, int32_t RealPrecision>
00096 AAX_CLogTaperDelegate<T, RealPrecision>::AAX_CLogTaperDelegate(T minValue, T maxValue) :
00097     AAX_ITaperDelegate<T>(),
00098     mMinValue(minValue),
00099     mMaxValue(maxValue)
00100 {
00101 }
00102
00103 template <typename T, int32_t RealPrecision>
00104 AAX_CLogTaperDelegate<T, RealPrecision>*    AAX_CLogTaperDelegate<T, RealPrecision>::Clone() const
00105 {
00106     return new AAX_CLogTaperDelegate(*this);
00107 }
00108
00109 template <typename T, int32_t RealPrecision>
00110 T        AAX_CLogTaperDelegate<T, RealPrecision>::ConstrainRealValue(T value) const
00111 {
00112     if (mMinValue == mMaxValue)
00113         return mMinValue;
00114     if (RealPrecision)
00115         value = Round(value);           //reduce the precision to get proper rounding behavior with
00116     integers.
00117     const T& highValue = mMaxValue > mMinValue ? mMaxValue : mMinValue;
00118     const T& lowValue = mMaxValue > mMinValue ? mMinValue : mMaxValue;
00119     if (value > highValue)
00120         return highValue;
00121     if (value < lowValue)
00122         return lowValue;
00123     return value;
00124 }
00125
00126 template <typename T, int32_t RealPrecision>
00127 T        AAX_CLogTaperDelegate<T, RealPrecision>::NormalizedToReal(double normalizedValue) const
00128 {
00129     double minLog = AAX::SafeLog(double(mMinValue));
00130     double maxLog = AAX::SafeLog(double(mMaxValue));
00131     double doubleRealValue = exp(normalizedValue * (maxLog - minLog) + minLog);
00132     T realValue = (T) doubleRealValue;
00133     return ConstrainRealValue(realValue);
00134 }
00135
00136 template <typename T, int32_t RealPrecision>
00137 double   AAX_CLogTaperDelegate<T, RealPrecision>::RealToNormalized(T realValue) const
00138 {
00139     double minLog = AAX::SafeLog(double(mMinValue));
00140     double maxLog = AAX::SafeLog(double(mMaxValue));
00141     realValue = ConstrainRealValue(realValue);
00142     double normalizedValue = (maxLog == minLog) ? 0.5 : (AAX::SafeLog(double(realValue)) - minLog) /
00143     (maxLog - minLog);
00144     return normalizedValue;
00145 }
00146
00147 #endif // AAX_CLOGTAPERDELEGATE_H

```

15.89 AAX_CMutex.h File Reference

15.89.1 Description

Mutex.

Classes

- class [AAX_CMutex](#)
Mutex with try lock functionality.
- class [AAX_StLock_Guard](#)
Helper class for working with mutex.

15.90 AAX_CMutex.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00013 #ifndef AAX_CMUTEX_H
00014 #define AAX_CMUTEX_H
00015
00022 /*=====*/
00023
00026 class AAX_CMutex
00027 {
00028 public:
00029     AAX_CMutex();
00030     ~AAX_CMutex();
00031
00032     bool Lock();
00033     void Unlock();
00034     bool Try_Lock();
00035
00036 private:
00037     AAX_CMutex(const AAX_CMutex&);
00038     AAX_CMutex& operator=(const AAX_CMutex&);
00039
00040     typedef struct opaque_aax_mutex_t * aax_mutex_t;
00041     aax_mutex_t mMutex;
00042 };
00043
00046 class AAX_StLock_Guard
00047 {
00048 public:
00049     explicit AAX_StLock_Guard(AAX_CMutex& iMutex) : mMutex(iMutex) { mNeedsUnlock = mMutex.Lock(); }
00050     ~AAX_StLock_Guard() { if (mNeedsUnlock) mMutex.Unlock(); }
00051
00052 private:
00053     AAX_StLock_Guard(AAX_StLock_Guard const&);
00054     AAX_StLock_Guard& operator=(AAX_StLock_Guard const&);
00055
00056     AAX_CMutex & mMutex;
00057     bool mNeedsUnlock;
00058 };
00059
00060 #endif // AAX_CMUTEX_H
00061
```

15.91 AAX_CNumberDisplayDelegate.h File Reference

```
#include "AAX_IDisplayDelegate.h"
#include "AAX_CString.h"
```


15.91.1 Description

A number display delegate.

Classes

- class [AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >](#)
A numeric display format conforming to [AAX_IDisplayDelegate](#).

15.92 AAX_CNumberDisplayDelegate.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CNUMBERDISPLAYDELEGATE_H
00023 #define AAX_CNUMBERDISPLAYDELEGATE_H
00024
00025 #include "AAX_IDisplayDelegate.h"
00026 #include "AAX_CString.h"
00027
00028
00038 template <typename T, uint32_t Precision=2, uint32_t SpaceAfter=0>
00039 class AAX_CNumberDisplayDelegate : public AAX_IDisplayDelegate<T>
00040 {
00041 public:
00042     //Virtual Overrides
00043     AAX_CNumberDisplayDelegate* Clone() const AAX_OVERRIDE;
00044     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00045     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const
00046     AAX_OVERRIDE;
00046     bool StringToValue(const AAX_CString& valueString, T* value) const AAX_OVERRIDE;
00047 };
00048
00049
00050
00051
00052 template <typename T, uint32_t Precision, uint32_t SpaceAfter>
00053 AAX_CNumberDisplayDelegate<T,Precision,SpaceAfter>*
00054 AAX_CNumberDisplayDelegate<T,Precision,SpaceAfter>::Clone() const
00055 {
00056     return new AAX_CNumberDisplayDelegate(*this);
00057 }
00058
00059 template <typename T, uint32_t Precision, uint32_t SpaceAfter>
00059 bool AAX_CNumberDisplayDelegate<T,Precision,SpaceAfter>::ValueToString(T value, AAX_CString*
00060 valueString) const
00061 {
00062     valueString->Clear();
00063     valueString->AppendNumber(value, Precision);
00064     if (SpaceAfter != 0)
00065         valueString->Append(" "); //Added a space after the number for easier display of units.
00066     return true;
00067 }
00068
00069 template <typename T, uint32_t Precision, uint32_t SpaceAfter>
00069 bool AAX_CNumberDisplayDelegate<T,Precision,SpaceAfter>::ValueToString(T value, int32_t
00070 maxNumChars, AAX_CString* valueString) const
00071 {
00072     valueString->Clear();
00073     valueString->AppendNumber(value, Precision);
00074     uint32_t strlen = valueString->Length();
```

```

00074     const uint32_t maxNumCharsUnsigned = (0 <= maxNumChars) ? static_cast<uint32_t>(maxNumChars) : 0;
00075     if (strlen > maxNumCharsUnsigned)
00076     {
00077         valueString->Erase(maxNumCharsUnsigned, strlen-maxNumCharsUnsigned);
00078         strlen = valueString->Length();
00079     }
00080
00081     if ( 0 < maxNumCharsUnsigned && strlen == maxNumCharsUnsigned &&
        (*valueString)[maxNumCharsUnsigned-1] == '.') //<DMT> Edge case when the decimal point is the last
        character, we probably shouldn't show it.
    {
00082         valueString->Erase(maxNumCharsUnsigned-1, 1);
00083         strlen = valueString->Length();
00084     }
00085
00086
00087     if ((SpaceAfter != 0) && (maxNumCharsUnsigned > strlen) && (maxNumCharsUnsigned-strlen > 2))
        //<DMT> Kind of a random threshold for dropping the space after, but seems reasonable for our control
        surfaces. (allows dB and Unit prefixes)
    {
00088         valueString->Append(" "); //Added a space after the number for easier display of units.
00089         return true;
00090     }
00091
00092 template <typename T, uint32_t Precision, uint32_t SpaceAfter>
00093 bool    AAX_CNumberDisplayDelegate<T,Precision,SpaceAfter>::StringValue(const AAX_CString&
        valueString, T* value) const
00094 {
00095     double dValue;
00096     if (valueString.ToDouble(&dValue))
00097     {
00098         *value = static_cast<T>(dValue);
00099         return true;
00100     }
00101     *value = 0;
00102     return false;
00103 }
00104
00105
00106
00107
00108 #endif //AAX_CNUMBERDISPLAYDELEGATE_H

```

15.93 AAX_CommonConversions.h File Reference

```

#include <math.h>
#include "AAX.h"

```

Functions

- double [GainToDB](#) (double aGain)
Convert Gain to dB.
- double [DBToGain](#) (double dB)
Convert dB to Gain.
- double [LongToDouble](#) (int32_t aLong)
Convert Long to Double.
- int32_t [DoubleToLong](#) (double aDouble)
convert floating point equivalent back to int32_t
- int32_t [DoubleToDSPCoef](#) (double d, double max=k56kFloatPosMax, double min=k56kFloatNegMax)
Convert Double to DSPCoef.
- double [DSPCoefToDouble](#) (int32_t c, int32_t max=k56kFracPosMax, int32_t min=k56kFracNegMax)
Convert DSPCoef to Double.
- double [ThirtyTwoBitDSPCoefToDouble](#) (int32_t c)
ThirtyTwoBitDSPCoefToDouble.
- int32_t [DoubleTo32BitDSPCoefRnd](#) (double d)
DoubleTo32BitDSPCoefRnd.
- int32_t [DoubleTo32BitDSPCoef](#) (double d)
- int32_t [DoubleToDSPCoefRnd](#) (double d, double max, double min)

Variables

- `const int32_t k32BitPosMax` = 0x7FFFFFFF
- `const int32_t k32BitAbsMax` = 0x80000000
- `const int32_t k32BitNegMax` = 0x80000000
- `const int32_t k56kFracPosMax` = 0x007FFFFF
- `const int32_t k56kFracAbsMax` = 0x00800000
- `const int32_t k56kFracHalf` = 0x00400000
- `const int32_t k56kFracNegOne` = 0xFF800000
- `const int32_t k56kFracNegMax` = `k56kFracNegOne`
- `const int32_t k56kFracZero` = 0x00000000
- `const double kOneOver56kFracAbsMax` = 1.0/double(`k56kFracAbsMax`)
- `const double k56kFloatPosMax` = double(`k56kFracPosMax`)/double(`k56kFracAbsMax`)
- `const double k56kFloatNegMax` = -1.0
- `const double kNeg144DB` = -144.0
- `const double kNeg144Gain` = 6.3095734448019324943436013662234e-8

15.93.1 Function Documentation

15.93.1.1 GainToDB()

```
double GainToDB (
    double aGain ) [inline]
```

Convert Gain to dB.

Todo This should be incorporated into parameters' tapers and not called separately

References [kNeg144DB](#).

15.93.1.2 DBToGain()

```
double DBToGain (
    double dB ) [inline]
```

Convert dB to Gain.

Todo This should be incorporated into parameters' tapers and not called separately

15.93.1.3 LongToDouble()

```
double LongToDouble (
    int32_t aLong ) [inline]
```

Convert Long to Double.

LongToDouble: convert 24 bit fixed point in a int32_t to floating point equivalent

References [k56kFracNegMax](#), [k56kFracPosMax](#), and [kOneOver56kFracAbsMax](#).

15.93.1.4 DoubleToLong()

```
int32_t DoubleToLong (
    double aDouble )
```

convert floating point equivalent back to int32_t

15.93.1.5 DoubleToDSPCoef()

```
int32_t DoubleToDSPCoef (
    double d,
    double max = k56kFloatPosMax,
    double min = k56kFloatNegMax ) [inline]
```

Convert Double to DSPCoef.

References [k56kFracAbsMax](#), [k56kFracNegMax](#), and [k56kFracPosMax](#).

Referenced by [DoubleTo32BitDSPCoefRnd\(\)](#).

Here is the caller graph for this function:



15.93.1.6 DSPCoefToDouble()

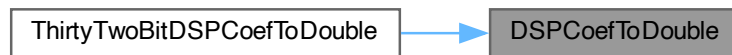
```
double DSPCoefToDouble (
    int32_t c,
    int32_t max = k56kFracPosMax,
    int32_t min = k56kFracNegMax ) [inline]
```

Convert DSPCoef to Double.

References [k56kFracNegMax](#), [k56kFracPosMax](#), and [kOneOver56kFracAbsMax](#).

Referenced by [ThirtyTwoBitDSPCoefToDouble\(\)](#).

Here is the caller graph for this function:



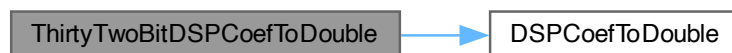
15.93.1.7 ThirtyTwoBitDSPCoefToDouble()

```
double ThirtyTwoBitDSPCoefToDouble (
    int32_t c ) [inline]
```

ThirtyTwoBitDSPCoefToDouble.

References [DSPCoefToDouble\(\)](#), [k32BitNegMax](#), and [k32BitPosMax](#).

Here is the call graph for this function:



15.93.1.8 DoubleTo32BitDSPCoefRnd()

```
int32_t DoubleTo32BitDSPCoefRnd (
    double d ) [inline]
```

DoubleTo32BitDSPCoefRnd.

References [DoubleToDSPCoef\(\)](#), [k32BitNegMax](#), and [k32BitPosMax](#).

Here is the call graph for this function:



15.93.1.9 DoubleTo32BitDSPCoef()

```
int32_t DoubleTo32BitDSPCoef (
    double d )
```

15.93.1.10 DoubleToDSPCoefRnd()

```
int32_t DoubleToDSPCoefRnd (
    double d,
    double max,
    double min )
```

15.93.2 Variable Documentation

15.93.2.1 k32BitPosMax

```
const int32_t k32BitPosMax = 0x7FFFFFFF
```

Referenced by [DoubleTo32BitDSPCoefRnd\(\)](#), and [ThirtyTwoBitDSPCoefToDouble\(\)](#).

15.93.2.2 k32BitAbsMax

```
const int32_t k32BitAbsMax = 0x80000000
```

15.93.2.3 k32BitNegMax

```
const int32_t k32BitNegMax = 0x80000000
```

Referenced by [DoubleTo32BitDSPCoefRnd\(\)](#), and [ThirtyTwoBitDSPCoefToDouble\(\)](#).

15.93.2.4 k56kFracPosMax

```
const int32_t k56kFracPosMax = 0x007FFFFF
```

Referenced by [DoubleToDSPCoef\(\)](#), [DSPCoefToDouble\(\)](#), and [LongToDouble\(\)](#).

15.93.2.5 k56kFracAbsMax

```
const int32_t k56kFracAbsMax = 0x00800000
```

Referenced by [DoubleToDSPCoef\(\)](#).

15.93.2.6 k56kFracHalf

```
const int32_t k56kFracHalf = 0x00400000
```

15.93.2.7 k56kFracNegOne

```
const int32_t k56kFracNegOne = 0xFF800000
```

15.93.2.8 k56kFracNegMax

```
const int32_t k56kFracNegMax = k56kFracNegOne
```

Referenced by [DoubleToDSPCoef\(\)](#), [DSPCoefToDouble\(\)](#), and [LongToDouble\(\)](#).

15.93.2.9 k56kFracZero

```
const int32_t k56kFracZero = 0x00000000
```

15.93.2.10 kOneOver56kFracAbsMax

```
const double kOneOver56kFracAbsMax = 1.0/double(k56kFracAbsMax)
```

Referenced by [DSPCoefToDouble\(\)](#), and [LongToDouble\(\)](#).

15.93.2.11 k56kFloatPosMax

```
const double k56kFloatPosMax = double(k56kFracPosMax)/double(k56kFracAbsMax)
```

15.93.2.12 k56kFloatNegMax

```
const double k56kFloatNegMax = -1.0
```

15.93.2.13 kNeg144DB

```
const double kNeg144DB = -144.0
```

Referenced by [GainToDB\(\)](#).

15.93.2.14 kNeg144Gain

```
const double kNeg144Gain = 6.3095734448019324943436013662234e-8
```


15.94 AAX_CommonConversions.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2014-2015, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012
00017  /*=====*/
00018
00019
00020  #ifndef AAX_COMMONCONVERSIONS_H
00021  #define AAX_COMMONCONVERSIONS_H
00022
00023  #include <math.h>
00024  #include "AAX.h"
00025
00026
00027  const int32_t k32BitPosMax          = 0x7FFFFFFF;
00028  const int32_t k32BitAbsMax          = 0x80000000;
00029  const int32_t k32BitNegMax          = 0x80000000;
00030
00031  const int32_t k56kFracPosMax        = 0x007FFFFF; // Positive Max Value
00032  const int32_t k56kFracAbsMax        = 0x00800000; // Absolute Max Value. Essentially negative one
00033  // without the sign extension.
00034  const int32_t k56kFracHalf          = 0x00400000;
00035  const int32_t k56kFracNegOne        = 0xFF800000; //Note sign extension!!!
00036  const int32_t k56kFracNegMax        = k56kFracNegOne; //Note sign extension!!!
00037  const int32_t k56kFracZero          = 0x00000000;
00038
00038  const double kOneOver56kFracAbsMax = 1.0/double(k56kFracAbsMax);
00039  const double k56kFloatPosMax       = double(k56kFracPosMax)/double(k56kFracAbsMax); //56k Max
00040  // value represented in floating point format.
00041  const double k56kFloatNegMax       = -1.0; //56k Min value represented in floating point format.
00042  const double kNeg144DB              = -144.0;
00043  const double kNeg144Gain            = 6.3095734448019324943436013662234e-8; //pow(10.0, kNeg144DB /
00044  // 20.0);
00045
00046  inline double GainToDB(double aGain)
00047  {
00048      if (aGain == 0.0)
00049          return kNeg144DB;
00050      else
00051      {
00052          double dB;
00053
00054          dB = log10(aGain) * 20.0;
00055
00056          if (dB < kNeg144DB)
00057              dB = kNeg144DB;
00058          return (dB); // convert factor to dB
00059      }
00060  }
00061
00062  inline double DBToGain(double dB)
00063  {
00064      return pow(10.0, dB / 20.0);
00065  }
00066
00067  inline double LongToDouble (int32_t aLong)
00068  {
00069      if (aLong > k56kFracPosMax)
00070          aLong = k56kFracPosMax;
00071      else if (aLong < k56kFracNegMax)
00072          aLong = k56kFracNegMax;
00073      return (double(aLong) * kOneOver56kFracAbsMax);
00074  }
00075
00076  int32_t DoubleToLong (double aDouble);
00077
00078  inline int32_t DoubleToDSPCoef(double d, double max = k56kFloatPosMax, double min = k56kFloatNegMax)
00079  {
00080      if (d >= max) // k56kFloatPosMax unless specified by the caller
00081      {
00082          return k56kFracPosMax;
00083      };
00084      if (d < min) // k56kFloatNegMax unless specified by the caller
00085      {

```

```

00103         return k56kFracNegMax;
00104     }
00105     return static_cast<int32_t>(d*k56kFracAbsMax);
00106 }
00107
00110 inline double DSPCoefToDouble(int32_t c, int32_t max = k56kFracPosMax, int32_t min = k56kFracNegMax)
00111 {
00112     if (c > max) // k56kFracPosMax unless specified by the caller
00113         c = k56kFracPosMax;
00114     else if (c < min) // k56kFracNegMax unless specified by the caller
00115         c = k56kFracNegMax;
00116     return (double(c) * kOneOver56kFracAbsMax);
00117 }
00118
00121 inline double ThirtyTwoBitDSPCoefToDouble(int32_t c)
00122 {
00123     return DSPCoefToDouble(c, k32BitPosMax, k32BitNegMax);
00124 }
00125
00128 inline int32_t DoubleTo32BitDSPCoefRnd(double d)
00129 {
00130     return DoubleToDSPCoef(d, k32BitPosMax, k32BitNegMax);
00131 }
00132
00133 int32_t DoubleTo32BitDSPCoef(double d);
00134 int32_t DoubleToDSPCoefRnd(double d, double max, double min);
00135
00136 #endif // AAX_COMMONCONVERSIONS_H

```

15.95 AAX_CPacketDispatcher.h File Reference

```

#include "AAX.h"
#include "AAX_IController.h"
#include "AAX_CMutex.h"
#include <string>
#include <map>

```

15.95.1 Description

Helper classes related to posting AAX packets and handling parameter update events.

Classes

- class [AAX_CPacket](#)
Container for packet-related data.
- struct [AAX_IPacketHandler](#)
Callback container used by [AAX_CPacketDispatcher](#).
- class [AAX_CPacketHandler< TWorker >](#)
Callback container used by [AAX_CPacketDispatcher](#).
- class [AAX_CPacketDispatcher](#)
Helper class for managing AAX packet posting.

15.96 AAX_CPacketDispatcher.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00019  /*=====*/
00020
00021
00022  #ifndef AAX_CPACKETDISPATCHER_H
00023  #define AAX_CPACKETDISPATCHER_H
00024
00025  #include "AAX.h"
00026  #include "AAX_IController.h"
00027  #include "AAX_CMutex.h"
00028
00029  #include <string>
00030  #include <map>
00031
00032
00041  class AAX_CPacket
00042  {
00043  public:
00044      AAX_CPacket(AAX_CFieldIndex inFieldIndex) : mID(inFieldIndex), mDirty(true), mDataSize(0) {}
00045      ~AAX_CPacket() {}
00046
00047      template<typename DataType>
00048      DataType* GetPtr()
00049      {
00050          mDataSize = sizeof(DataType);
00051          void * data = mPacketData.Get(mDataSize);
00052          return reinterpret_cast<DataType*> (data);
00053      }
00054
00055      void SetDirty(bool iDirty) { mDirty = iDirty; };
00056      bool IsDirty() const { return mDirty; };
00057
00058      AAX_CFieldIndex GetID() const { return mID; };
00059      uint32_t GetSize() const { return mDataSize; }
00060
00061  private:
00062      AAX_CFieldIndex mID;
00063      bool mDirty;
00064      uint32_t mDataSize;
00065
00066  private:
00067      struct SPacketData
00068      {
00069      public:
00070          SPacketData();
00071          ~SPacketData();
00072          const void* Get() const;
00073          void* Get(size_t newSize) const;
00074      private:
00075          mutable void* mData;
00076      } mPacketData;
00077  };
00078
00079  // GetPtr() specialization for void*
00080  template <>
00081  inline const void*
00082  AAX_CPacket::GetPtr<const void*>()
00083  {
00084      return mPacketData.Get();
00085  }
00086
00087
00090  struct AAX_IPacketHandler
00091  {
00092      virtual ~AAX_IPacketHandler() {};
00093      virtual AAX_IPacketHandler* Clone() const = 0;
00094      virtual AAX_Result Call(AAX_CParamID inParamID, AAX_CPacket& ioPacket) const = 0;
00095  };
00096
00099  template<class TWorker>
00100  class AAX_CPacketHandler : public AAX_IPacketHandler

```

```

00101 {
00102     typedef AAX_Result(TWorker::*fPt2Fn)(AAX_CPacket&);
00103     typedef AAX_Result(TWorker::*fPt2FnEx)(AAX_CParamID, AAX_CPacket&);
00104
00105 public:
00106     AAX_CPacketHandler( TWorker* iPt2Object, fPt2Fn infPt )
00107         : pt2Object(iPt2Object), fpt(infPt), fptEx(NULL) {}
00108
00109     AAX_CPacketHandler( TWorker* iPt2Object, fPt2FnEx infPt )
00110         : pt2Object(iPt2Object), fpt(NULL), fptEx(infPt) {}
00111
00112     AAX_IPacketHandler* Clone() const
00113     {
00114         return new AAX_CPacketHandler(*this);
00115     }
00116
00117     AAX_Result Call( AAX_CParamID inParamID, AAX_CPacket& ioPacket ) const
00118     {
00119         if (fptEx)
00120             return (*pt2Object.*fptEx)( inParamID, ioPacket);
00121         else if (fpt)
00122             return (*pt2Object.*fpt)( ioPacket);
00123         else
00124             return AAX_ERROR_NULL_OBJECT;
00125     }
00126
00127 protected:
00128     TWorker *   pt2Object; // pointer to object
00129     fPt2Fn      fpt ;      // pointer to member function
00130     fPt2FnEx    fptEx ;    // pointer to member function
00131 };
00132
00133
00134 class AAX_IEffectParameters;
00135
00149 class AAX_CPacketDispatcher
00150 {
00151     typedef std::map<AAX_CFieldIndex, AAX_CPacket*> PacketsHolder;
00152     typedef std::multimap<std::string, std::pair<AAX_CPacket*, AAX_IPacketHandler*> >
00153     PacketsHandlersMap;
00154 public:
00155     AAX_CPacketDispatcher();
00156     ~AAX_CPacketDispatcher();
00157
00158     void Initialize( AAX_IController* iPlugIn, AAX_IEffectParameters* iEffectParameters);
00159
00160     AAX_Result RegisterPacket( AAX_CParamID paramID, AAX_CFieldIndex portID, const AAX_IPacketHandler*
00161     iHandler);
00162
00163     template <class TWorker, typename Func>
00164     AAX_Result RegisterPacket( AAX_CParamID paramID, AAX_CFieldIndex portID,
00165                               TWorker* iPt2Object, Func infPt)
00166     {
00167         AAX_CPacketHandler<TWorker> handler(iPt2Object, infPt);
00168         return RegisterPacket(paramID, portID, &handler);
00169     }
00170
00171     AAX_Result RegisterPacket( AAX_CParamID paramID, AAX_CFieldIndex portID)
00172     {
00173         AAX_CPacketHandler<AAX_CPacketDispatcher> handler(this,
00174     &AAX_CPacketDispatcher::GenerateSingleValuePacket);
00175         return RegisterPacket(paramID, portID, &handler);
00176     }
00177
00178     AAX_Result SetDirty(AAX_CParamID paramID, bool iDirty = true);
00179
00180     AAX_Result Dispatch();
00181
00182     AAX_Result GenerateSingleValuePacket( AAX_CParamID iParam, AAX_CPacket& ioPacket);
00183
00184 private:
00185     PacketsHolder mPacketsHolder;
00186     PacketsHandlersMap mPacketsHandlers;
00187     AAX_IController* mController;
00188     AAX_IEffectParameters* mEffectParameters;
00189
00190     AAX_CMutex mLockGuard;
00191 };
00192 #endif // AAX_CPACKETDISPATCHER_H

```

15.97 AAX_CParameter.h File Reference

```
#include "AAX_Assert.h"
#include "AAX_IParameter.h"
#include "AAX_ITaperDelegate.h"
#include "AAX_IDisplayDelegate.h"
#include "AAX_IAutomationDelegate.h"
#include "AAX_CString.h"
#include <cstring>
#include <list>
#include <map>
```

15.97.1 Description

Generic implementation of an [AAX_IParameter](#).

Classes

- class [AAX_CParameterValue< T >](#)
Concrete implementation of [AAX_IParameterValue](#).
- class [AAX_CParameter< T >](#)
Generic implementation of an [AAX_IParameter](#).
- class [AAX_CStatelessParameter](#)
A stateless parameter implementation.

15.98 AAX_CParameter.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011 */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CPARAMETER_H
00023 #define AAX_CPARAMETER_H
00024
00025 #include "AAX_Assert.h"
00026 #include "AAX_IParameter.h"
00027 #include "AAX_ITaperDelegate.h"
00028 #include "AAX_IDisplayDelegate.h"
00029 #include "AAX_IAutomationDelegate.h"
00030 #include "AAX_CString.h" //concrete class required for name.
00031
00032 #include <cstring>
00033 #include <list>
00034 #include <map>
00035
00036
00038 #if 0
00039 #pragma mark -
00040 #endif
```

```

00042
00043
00050 template <typename T>
00051 class AAX_CParameterValue : public AAX_IParameterValue
00052 {
00053 public:
00054     enum Defaults {
00055         eParameterDefaultMaxIdentifierSize = kAAX_ParameterIdentifierMaxSize,
00056         eParameterDefaultMaxIdentifierLength = eParameterDefaultMaxIdentifierSize - 1 // NULL
00057         terminated
00058     };
00059 public:
00060     AAX_DEFAULT_DTOR_OVERRIDE(AAX_CParameterValue);
00061
00062     AAX_DEFAULT_MOVE_CTOR(AAX_CParameterValue);
00063     AAX_DEFAULT_MOVE_OPER(AAX_CParameterValue);
00064
00065     AAX_DELETE(AAX_CParameterValue& operator=(const AAX_CParameterValue&));
00066
00074     explicit AAX_CParameterValue(AAX_CParamID identifier);
00075
00083     explicit AAX_CParameterValue(AAX_CParamID identifier, const T& value);
00084
00087     explicit AAX_CParameterValue(const AAX_CParameterValue<T>& other);
00088
00089 public: // AAX_CParameterValue<T> implementation
00092     const T& Get() const { return mValue; }
00095     void Set(const T& inValue) { mValue = inValue; }
00096
00097 public: // AAX_IParameterValue implementation
00098
00099     AAX_IParameterValue* Clone() const AAX_OVERRIDE { return new AAX_CParameterValue<T>(*this); }
00100     AAX_CParamID Identifier() const AAX_OVERRIDE { return mIdentifier; }
00101
00106     bool GetValueAsBool(bool* value) const AAX_OVERRIDE;
00107     bool GetValueAsInt32(int32_t* value) const AAX_OVERRIDE;
00108     bool GetValueAsFloat(float* value) const AAX_OVERRIDE;
00109     bool GetValueAsDouble(double* value) const AAX_OVERRIDE;
00110     bool GetValueAsString(AAX_IString* value) const AAX_OVERRIDE;
00111
00112 private:
00114     void InitIdentifier(const char* inIdentifier);
00115
00116 private:
00117     char mIdentifier[eParameterDefaultMaxIdentifierSize];
00118     T mValue;
00119 };
00120
00121
00122
00123
00124 template <typename T>
00125 AAX_CParameterValue<T>::AAX_CParameterValue(AAX_CParamID identifier)
00126 : mValue()
00127 {
00128     InitIdentifier(identifier);
00129 }
00130
00131 template <typename T>
00132 AAX_CParameterValue<T>::AAX_CParameterValue(AAX_CParamID identifier, const T& value)
00133 : mValue(value)
00134 {
00135     InitIdentifier(identifier);
00136 }
00137
00138 template <typename T>
00139 AAX_CParameterValue<T>::AAX_CParameterValue(const AAX_CParameterValue<T>& other)
00140 : mValue(other.mValue)
00141 {
00142     InitIdentifier(other.mIdentifier);
00143 }
00144
00145 template<typename T>
00146 bool AAX_CParameterValue<T>::GetValueAsBool(bool* /*value*/) const
00147 {
00148     return false;
00149 }
00150
00151 template<>
00152 bool AAX_CParameterValue<bool>::GetValueAsBool(bool* value) const;
00153
00154 template<typename T>
00155 bool AAX_CParameterValue<T>::GetValueAsInt32(int32_t* /*value*/) const
00156 {
00157     return false;
00158 }
00159
00160 template<>

```

```

00160 bool          AAX_CParameterValue<int32_t>::GetValueAsInt32(int32_t* value) const;
00161
00162 template<typename T>
00163 bool          AAX_CParameterValue<T>::GetValueAsFloat(float* /*value*/) const
00164 {
00165     return false;
00166 }
00167 template<>
00168 bool          AAX_CParameterValue<float>::GetValueAsFloat(float* value) const;
00169
00170 template<typename T>
00171 bool          AAX_CParameterValue<T>::GetValueAsDouble(double* /*value*/) const
00172 {
00173     return false;
00174 }
00175 template<>
00176 bool          AAX_CParameterValue<double>::GetValueAsDouble(double* value) const;
00177
00178 template<typename T>
00179 bool          AAX_CParameterValue<T>::GetValueAsString(AAX_IString* /*value*/) const
00180 {
00181     return false;
00182 }
00183 template<>
00184 bool          AAX_CParameterValue<AAX_CString>::GetValueAsString(AAX_IString* value) const;
00185
00186 template<typename T>
00187 void AAX_CParameterValue<T>::InitIdentifier(const char *inIdentifier)
00188 {
00189
00190     const size_t len = strlen(inIdentifier);
00191     AAX_ASSERT(len < eParameterDefaultMaxIdentifierSize);
00192     if (len < eParameterDefaultMaxIdentifierSize)
00193     {
00194         std::strncpy(mIdentifier, inIdentifier, 1+len);
00195         mIdentifier[len] = 0;
00196     }
00197     else
00198     {
00199         std::strncpy(mIdentifier, inIdentifier, eParameterDefaultMaxIdentifierLength);
00200         mIdentifier[eParameterDefaultMaxIdentifierLength] = 0;
00201     }
00202 }
00203
00204
00206 #if 0
00207 #pragma mark -
00208 #endif
00210
00212
00233 template <typename T>
00234 class AAX_CParameter : public AAX_IParameter
00235 {
00236 public:
00237
00238     enum Type {
00239         eParameterTypeUndefined = 0,
00240         eParameterTypeBool = 1,
00241         eParameterTypeInt32 = 2,
00242         eParameterTypeFloat = 3,
00243         eParameterTypeCustom = 4
00244     };
00245
00246     enum Defaults {
00247         eParameterDefaultNumStepsDiscrete = 2,
00248         eParameterDefaultNumStepsContinuous = 128
00249     };
00250
00281     AAX_CParameter(AAX_CParamID identifier, const AAX_IString& name, T defaultValue, const
AAX_ITaperDelegate<T>& taperDelegate, const AAX_IDisplayDelegate<T>& displayDelegate, bool
automatable=false);
00282
00289     AAX_CParameter(const AAX_IString& identifier, const AAX_IString& name, T defaultValue, const
AAX_ITaperDelegate<T>& taperDelegate, const AAX_IDisplayDelegate<T>& displayDelegate, bool
automatable=false);
00290
00300     AAX_CParameter(const AAX_IString& identifier, const AAX_IString& name, T defaultValue, bool
automatable=false);
00301
00312     AAX_CParameter(const AAX_IString& identifier, const AAX_IString& name, bool automatable=false);
00313
00315     AAX_DEFAULT_MOVE_CTOR(AAX_CParameter);
00316     AAX_DEFAULT_MOVE_OPER(AAX_CParameter);
00317
00319     AAX_DELETE(AAX_CParameter());
00320     AAX_DELETE(AAX_CParameter(const AAX_CParameter& other));
00321     AAX_DELETE(AAX_CParameter& operator= (const AAX_CParameter& other));

```

```

00322
00327 ~AAX_CParameter() AAX_OVERRIDE;
00328
00329 AAX_IParameterValue* CloneValue() const AAX_OVERRIDE;
00330
00335 AAX_CParamID Identifier() const AAX_OVERRIDE;
00336 void SetName(const AAX_CString& name) AAX_OVERRIDE;
00337 const AAX_CString& Name() const AAX_OVERRIDE;
00338 void AddShortenedName(const AAX_CString& name) AAX_OVERRIDE;
00339 const AAX_CString& ShortenedName(int32_t iNumCharacters) const AAX_OVERRIDE;
00340 void ClearShortenedNames() AAX_OVERRIDE;
00342
00347 void SetNormalizedDefaultValue(double normalizedDefault) AAX_OVERRIDE;
00348 double GetNormalizedDefaultValue() const AAX_OVERRIDE;
00349 void SetToDefaultValue() AAX_OVERRIDE;
00350 void SetNormalizedValue(double newNormalizedValue) AAX_OVERRIDE;
00351 double GetNormalizedValue() const AAX_OVERRIDE;
00352 void SetNumberOfSteps(uint32_t numSteps) AAX_OVERRIDE;
00353 uint32_t GetNumberOfSteps() const AAX_OVERRIDE;
00354 uint32_t GetStepValue() const AAX_OVERRIDE;
00355 double GetNormalizedValueFromStep(uint32_t iStep) const AAX_OVERRIDE;
00356 uint32_t GetStepValueFromNormalizedValue(double normalizedValue) const AAX_OVERRIDE;
00357 void SetStepValue(uint32_t iStep) AAX_OVERRIDE;
00358 void SetType(AAX_EParameterType iControlType) AAX_OVERRIDE;
00359 AAX_EParameterType GetType() const AAX_OVERRIDE;
00360 void SetOrientation(AAX_EParameterOrientation iOrientation) AAX_OVERRIDE;
00361 AAX_EParameterOrientation GetOrientation() const AAX_OVERRIDE;
00362 void SetTaperDelegate(AAX_ITaperDelegateBase& inTaperDelegate, bool
inPreserveValue=true) AAX_OVERRIDE;
00364
00369 void SetDisplayDelegate(AAX_IDisplayDelegateBase& inDisplayDelegate) AAX_OVERRIDE;
00370 bool GetValueString(AAX_CString* valueString) const AAX_OVERRIDE;
00371 bool GetValueString(int32_t iMaxNumChars, AAX_CString* valueString) const
AAX_OVERRIDE;
00372 bool GetNormalizedValueFromBool(bool value, double *normalizedValue) const
AAX_OVERRIDE;
00373 bool GetNormalizedValueFromInt32(int32_t value, double *normalizedValue) const
AAX_OVERRIDE;
00374 bool GetNormalizedValueFromFloat(float value, double *normalizedValue) const
AAX_OVERRIDE;
00375 bool GetNormalizedValueFromDouble(double value, double *normalizedValue) const
AAX_OVERRIDE;
00376 bool GetNormalizedValueFromString(const AAX_CString& valueString, double
*normalizedValue) const AAX_OVERRIDE;
00377 bool GetBoolFromNormalizedValue(double normalizedValue, bool* value) const
AAX_OVERRIDE;
00378 bool GetInt32FromNormalizedValue(double normalizedValue, int32_t* value) const
AAX_OVERRIDE;
00379 bool GetFloatFromNormalizedValue(double normalizedValue, float* value) const
AAX_OVERRIDE;
00380 bool GetDoubleFromNormalizedValue(double normalizedValue, double* value) const
AAX_OVERRIDE;
00381 bool GetStringFromNormalizedValue(double normalizedValue, AAX_CString& valueString)
const AAX_OVERRIDE;
00382 bool GetStringFromNormalizedValue(double normalizedValue, int32_t iMaxNumChars,
AAX_CString& valueString) const AAX_OVERRIDE;
00383 bool SetValueFromString(const AAX_CString& newValueString) AAX_OVERRIDE;
00385
00390 void SetAutomationDelegate(AAX_IAutomationDelegate * iAutomationDelegate)
AAX_OVERRIDE;
00391 bool Automatable() const AAX_OVERRIDE;
00392 void Touch() AAX_OVERRIDE;
00393 void Release() AAX_OVERRIDE;
00395
00400 bool GetValueAsBool(bool* value) const AAX_OVERRIDE;
00401 bool GetValueAsInt32(int32_t* value) const AAX_OVERRIDE;
00402 bool GetValueAsFloat(float* value) const AAX_OVERRIDE;
00403 bool GetValueAsDouble(double* value) const AAX_OVERRIDE;
00404 bool GetValueAsString(AAX_IString* value) const AAX_OVERRIDE;
00405 bool SetValueWithBool(bool value) AAX_OVERRIDE;
00406 bool SetValueWithInt32(int32_t value) AAX_OVERRIDE;
00407 bool SetValueWithFloat(float value) AAX_OVERRIDE;
00408 bool SetValueWithDouble(double value) AAX_OVERRIDE;
00409 bool SetValueWithString(const AAX_IString& value) AAX_OVERRIDE;
00411
00416 void UpdateNormalizedValue(double newNormalizedValue) AAX_OVERRIDE;
00418
00436 void SetValue(T newValue);
00443 T GetValue() const;
00452 void SetDefaultValue(T newDefaultValue);
00459 T GetDefaultValue() const;
00464 const AAX_ITaperDelegate<T>* TaperDelegate() const;
00469 const AAX_IDisplayDelegate<T>* DisplayDelegate() const;
00471
00472 protected:
00473 AAX_CStringAbbreviations mNames;
00474 bool mAutomatable;

```



```

00475     uint32_t                                mNumSteps;
00476     AAX_EParameterType                      mControlType;
00477     AAX_EParameterOrientation              mOrientation;
00478     AAX_ITaperDelegate<T> *                mTaperDelegate;
00479     AAX_IDisplayDelegate<T> *             mDisplayDelegate;
00480     AAX_IAutomationDelegate *             mAutomationDelegate;
00481     bool                                    mNeedNotify;
00482
00483     AAX_CParameterValue<T>                mValue;
00484     T                                       mDefaultValue;
00485
00486 private:
00487     void InitializeNumberOfSteps();
00488 };
00489
00490
00491
00492
00493 template <typename T>
00494 AAX_CParameter<T>::AAX_CParameter(AAX_CParamID identifier, const AAX_IString& name, T defaultValue,
    const AAX_ITaperDelegate<T>& taperDelegate, const AAX_IDisplayDelegate<T>& displayDelegate, bool
    automatable)
00495 : mNames(name)
00496 , mAutomatable(automatable)
00497 , mNumSteps(0) // Default set below for discrete/continuous
00498 , mControlType( AAX_eParameterType_Continuous )
00499 , mOrientation( AAX_eParameterOrientation_Default )
00500 , mTaperDelegate(taperDelegate.Clone())
00501 , mDisplayDelegate(displayDelegate.Clone())
00502 , mAutomationDelegate(0)
00503 , mNeedNotify(true)
00504 , mValue(identifier)
00505 , mDefaultValue(defaultValue)
00506 {
00507     this->InitializeNumberOfSteps();
00508     this->SetToDefaultValue();
00509 }
00510
00511 template <typename T>
00512 AAX_CParameter<T>::AAX_CParameter(const AAX_IString& identifier, const AAX_IString& name, T
    defaultValue, const AAX_ITaperDelegate<T>& taperDelegate, const AAX_IDisplayDelegate<T>&
    displayDelegate, bool automatable)
00513 : mNames(name)
00514 , mAutomatable(automatable)
00515 , mNumSteps(0) // Default set below for discrete/continuous
00516 , mControlType( AAX_eParameterType_Continuous )
00517 , mOrientation( AAX_eParameterOrientation_Default )
00518 , mTaperDelegate(taperDelegate.Clone())
00519 , mDisplayDelegate(displayDelegate.Clone())
00520 , mAutomationDelegate(0)
00521 , mNeedNotify(true)
00522 , mValue(identifier.Get())
00523 , mDefaultValue(defaultValue)
00524 {
00525     this->InitializeNumberOfSteps();
00526     this->SetToDefaultValue();
00527 }
00528
00529 template <typename T>
00530 AAX_CParameter<T>::AAX_CParameter(const AAX_IString& identifier, const AAX_IString& name, T
    defaultValue, bool automatable)
00531 : mNames(name)
00532 , mAutomatable(automatable)
00533 , mNumSteps(0)
00534 , mControlType( AAX_eParameterType_Continuous )
00535 , mOrientation( AAX_eParameterOrientation_Default )
00536 , mTaperDelegate(NULL)
00537 , mDisplayDelegate(NULL)
00538 , mAutomationDelegate(NULL)
00539 , mNeedNotify(true)
00540 , mValue(identifier)
00541 , mDefaultValue(defaultValue)
00542 {
00543     this->InitializeNumberOfSteps();
00544     this->SetToDefaultValue();
00545 }
00546
00547 template <typename T>
00548 AAX_CParameter<T>::AAX_CParameter(const AAX_IString& identifier, const AAX_IString& name, bool
    automatable)
00549 : mNames(name)
00550 , mAutomatable(automatable)
00551 , mNumSteps(0)
00552 , mControlType( AAX_eParameterType_Continuous )
00553 , mOrientation( AAX_eParameterOrientation_Default )
00554 , mTaperDelegate(NULL)
00555 , mDisplayDelegate(NULL)
00556 , mAutomationDelegate(NULL)

```

```

00557 , mNeedNotify(true)
00558 , mValue(identifier)
00559 , mDefaultValue()
00560 {
00561     this->InitializeNumberOfSteps();
00562     this->SetToDefaultValue(); // WARNING: uninitialized default value
00563 }
00564
00565 template <typename T>
00566 AAX_CParameter<T>::~AAX_CParameter()
00567 {
00568     //Make sure to remove any registration with the token system.
00569     SetAutomationDelegate(0);
00570
00571     delete mTaperDelegate;
00572     mTaperDelegate = 0;
00573     delete mDisplayDelegate;
00574     mDisplayDelegate = 0;
00575 }
00576
00577 template <typename T>
00578 AAX_IParameValue* AAX_CParameter<T>::CloneValue() const
00579 {
00580     return new AAX_CParameterValue<T>(mValue);
00581 }
00582
00583 template <typename T>
00584 AAX_CParamID AAX_CParameter<T>::Identifier() const
00585 {
00586     return mValue.Identifier();
00587 }
00588
00589 template <typename T>
00590 void AAX_CParameter<T>::SetName(const AAX_CString& name)
00591 {
00592     mNames.SetPrimary(name);
00593     if (mAutomationDelegate) {
00594         mAutomationDelegate->ParameterNameChanged(this->Identifier());
00595     }
00596 }
00597
00598 template <typename T>
00599 const AAX_CString& AAX_CParameter<T>::Name() const
00600 {
00601     return mNames.Primary();
00602 }
00603
00604 template <typename T>
00605 void AAX_CParameter<T>::AddShortenedName(const AAX_CString& name)
00606 {
00607     mNames.Add(name);
00608 }
00609
00610 template <typename T>
00611 const AAX_CString& AAX_CParameter<T>::ShortenedName(int32_t iNumCharacters) const
00612 {
00613     return mNames.Get(iNumCharacters);
00614 }
00615
00616 template <typename T>
00617 void AAX_CParameter<T>::ClearShortenedNames()
00618 {
00619     mNames.Clear();
00620 }
00621
00622
00623
00624 template<typename T>
00625 void AAX_CParameter<T>::SetValue( T newValue )
00626 {
00627     double newNormalizedValue = mTaperDelegate->RealToNormalized(newValue);
00628
00629     // <DMT> Always go through the automation delegate even if the control isn't automatable to
    prevent fighting with other GUIs.
00630     // Somewhere back in the automation delegate, or elsewhere in the system, it will determine the
    differences in behavior surrounding
00631     // automation. The only reason that there wouldn't be an automation delegate is if this parameter
    has yet to be added to a
00632     // ParameterManager. Let's put the null value guards in place, just in case, and also for unit
    tests.
00633     if ( mAutomationDelegate )
00634     {
00635         //TODO: Create RAII utility class for touch/release
00636
00637         //Touch the control
00638         Touch();
00639

```

```

00640         //Send that token.
00641         mAutomationDelegate->PostSetValueRequest(Identifier(), newNormalizedValue );
00642
00643         //Release the control
00644         Release();
00645     }
00646     else
00647     {
00648         mNeedNotify = true;
00649
00650         // In the rare case that an automation delegate doesn't exist, lets still set the value. It's
possible that someone is trying to
00651         // set the new value before adding the parameter to a parametermanager.
00652         UpdateNormalizedValue(newNormalizedValue);
00653     }
00654 }
00655
00656 template <typename T>
00657 void AAX_CParameter<T>::UpdateNormalizedValue(double newNormalizedValue)
00658 {
00659     T newValue = mTaperDelegate->NormalizedToReal(newNormalizedValue);
00660     if (mNeedNotify || (mValue.Get() != newValue))
00661     {
00662         //Set the new value
00663         mValue.Set(newValue);
00664
00665         //<DMT> Always notify that the value has changed through the automation delegate to guarantee
that all control surfaces and other
00666         // GUIs get their values updated.
00667         if (mAutomationDelegate)
00668             mAutomationDelegate->PostCurrentValue(Identifier(), newNormalizedValue);
00669
00670         // clear flag
00671         mNeedNotify = false;
00672     }
00673 }
00674
00675 template <typename T>
00676 void AAX_CParameter<T>::InitializeNumberOfSteps()
00677 {
00678     if (mNumSteps == 0) // If no explicit number of steps has been set...
00679     {
00680         switch (mControlType)
00681         {
00682             case AAX_eParameterType_Discrete:
00683             {
00684                 // Discrete parameters default to binary unless
00685                 // otherwise specified
00686                 this->SetNumberOfSteps (eParameterDefaultNumStepsDiscrete);
00687                 break;
00688             }
00689             case AAX_eParameterType_Continuous:
00690             {
00691                 // Defaulting to 128 steps to match one full rotation of
00692                 // Command|8 and similar surfaces, which query the num
00693                 // steps to determine tick values for rotary encoders
00694                 this->SetNumberOfSteps (eParameterDefaultNumStepsContinuous);
00695                 break;
00696             }
00697             default:
00698             {
00699                 AAX_ASSERT (0); // Invalid type
00700                 break;
00701             }
00702         }
00703     }
00704 }
00705
00706 template<typename T>
00707 T AAX_CParameter<T>::GetValue() const
00708 {
00709     return mValue.Get();
00710 }
00711
00712 template<typename T>
00713 bool AAX_CParameter<T>::GetValueAsBool(bool* value) const
00714 {
00715     return mValue.GetValueAsBool(value);
00716 }
00717
00718 template<typename T>
00719 bool AAX_CParameter<T>::GetValueAsInt32(int32_t* value) const
00720 {
00721     return mValue.GetValueAsInt32(value);
00722 }
00723
00724

```

```

00725 template<typename T>
00726 bool      AAX_CParameter<T>::GetValueAsFloat(float* value) const
00727 {
00728     return mValue.GetValueAsFloat(value);
00729 }
00730
00731 template<typename T>
00732 bool      AAX_CParameter<T>::GetValueAsDouble(double* value) const
00733 {
00734     return mValue.GetValueAsDouble(value);
00735 }
00736
00737 template<typename T>
00738 bool      AAX_CParameter<T>::GetValueAsString(AAX_IString* value) const
00739 {
00740     bool result = false;
00741     if (value)
00742     {
00743         AAX_CString valueString;
00744         result = this->GetValueString(&valueString);
00745         if (true == result)
00746         {
00747             *value = valueString;
00748         }
00749     }
00750     return result;
00751 }
00752
00753 template<>
00754 bool      AAX_CParameter<AAX_CString>::GetValueAsString(AAX_IString* /*value*/) const;
00755
00756 template<typename T>
00757 bool      AAX_CParameter<T>::SetValueWithBool(bool /*value*/)
00758 {
00759     return false;
00760 }
00761
00762 template<>
00763 bool      AAX_CParameter<bool>::SetValueWithBool(bool value);
00764
00765 template<typename T>
00766 bool      AAX_CParameter<T>::SetValueWithInt32(int32_t /*value*/)
00767 {
00768     return false;
00769 }
00770
00771 template<>
00772 bool      AAX_CParameter<int32_t>::SetValueWithInt32(int32_t value);
00773
00774 template<typename T>
00775 bool      AAX_CParameter<T>::SetValueWithFloat(float /*value*/)
00776 {
00777     return false;
00778 }
00779
00780 template<>
00781 bool      AAX_CParameter<float>::SetValueWithFloat(float value);
00782
00783 template<typename T>
00784 bool      AAX_CParameter<T>::SetValueWithDouble(double /*value*/)
00785 {
00786     return false;
00787 }
00788
00789 template<>
00790 bool      AAX_CParameter<double>::SetValueWithDouble(double value);
00791
00792 template<typename T>
00793 bool      AAX_CParameter<T>::SetValueWithString(const AAX_IString& value)
00794 {
00795     const AAX_CString valueString(value);
00796     return this->SetValueFromString(valueString);
00797 }
00798
00799 template<>
00800 bool      AAX_CParameter<AAX_CString>::SetValueWithString(const AAX_IString& value);
00801
00802 template<typename T>
00803 void      AAX_CParameter<T>::SetNormalizedDefaultValue(double newNormalizedDefault)
00804 {
00805     T newDefaultValue = mTaperDelegate->NormalizedToReal(newNormalizedDefault);
00806     SetDefaultValue(newDefaultValue);
00807 }
00808
00809 template<typename T>
00810 double    AAX_CParameter<T>::GetNormalizedDefaultValue() const
00811 {
00812     double normalizedDefault = mTaperDelegate->RealToNormalized(mDefaultValue);
00813     return normalizedDefault;
00814 }

```

```

00812 template<typename T>
00813 void    AAX_CParameter<T>::SetDefaultValue(T newDefaultValue)
00814 {
00815     newDefaultValue = mTaperDelegate->ConstrainRealValue(newDefaultValue);
00816     mDefaultValue = newDefaultValue;
00817 }
00818
00819 template<typename T>
00820 T        AAX_CParameter<T>::GetDefaultValue() const
00821 {
00822     return mDefaultValue;
00823 }
00824
00825 template<typename T>
00826 void    AAX_CParameter<T>::SetToDefaultValue()
00827 {
00828     SetValue(mDefaultValue);
00829 }
00830
00831 template<typename T>
00832 void    AAX_CParameter<T>::SetNumberOfSteps(uint32_t numSteps)
00833 {
00834     AAX_ASSERT(0 < numSteps);
00835     if (0 < numSteps)
00836     {
00837         mNumSteps = numSteps;
00838     }
00839 }
00840
00841 template<typename T>
00842 uint32_t AAX_CParameter<T>::GetNumberOfSteps() const
00843 {
00844     return mNumSteps;
00845 }
00846
00847 template<typename T>
00848 uint32_t AAX_CParameter<T>::GetStepValue() const
00849 {
00850     return GetStepValueFromNormalizedValue(this->GetNormalizedValue());
00851 }
00852
00853 template<typename T>
00854 double   AAX_CParameter<T>::GetNormalizedValueFromStep(uint32_t iStep) const
00855 {
00856     double numSteps = (double) this->GetNumberOfSteps ();
00857     if ( numSteps < 2.0 )
00858         return 0.0;
00859
00860     double valuePerStep = 1.0 / ( numSteps - 1.0 );
00861     double value = valuePerStep * (double) iStep;
00862     if ( value < 0.0 )
00863         value = 0.0;
00864     else if ( value > 1.0 )
00865         value = 1.0;
00866
00867     return value;
00868 }
00869
00870 template<typename T>
00871 uint32_t AAX_CParameter<T>::GetStepValueFromNormalizedValue(double normalizedValue) const
00872 {
00873     double numSteps = (double) this->GetNumberOfSteps ();
00874     if ( numSteps < 2.0 )
00875         return 0;
00876
00877     double valuePerStep = 1.0 / ( numSteps - 1.0 );
00878     double curStep = ( normalizedValue / valuePerStep ) + 0.5;
00879     if ( curStep < 0.0 )
00880         curStep = 0.0;
00881     else if ( curStep > (double) ( numSteps - 1.0 ) )
00882         curStep = (double) ( numSteps - 1.0 );
00883
00884     return (uint32_t) curStep;
00885 }
00886
00887 template<typename T>
00888 void    AAX_CParameter<T>::SetStepValue(uint32_t iStep)
00889 {
00890     double numSteps = (double) this->GetNumberOfSteps ();
00891     if ( numSteps < 2.0 )
00892         return;
00893
00894     this->SetNormalizedValue ( GetNormalizedValueFromStep(iStep) );
00895 }
00896
00897 template<typename T>
00898 void    AAX_CParameter<T>::SetType(AAX_EParameterType iControlType)

```

```

00899 {
00900     mControlType = iControlType;
00901 }
00902
00903 template<typename T>
00904 AAX_EParameterType AAX_CParameter<T>::GetType() const
00905 {
00906     return mControlType;
00907 }
00908
00909 template<typename T>
00910 void AAX_CParameter<T>::SetOrientation(AAX_EParameterOrientation iOrientation)
00911 {
00912     mOrientation = iOrientation;
00913 }
00914
00915 template<typename T>
00916 AAX_EParameterOrientation AAX_CParameter<T>::GetOrientation() const
00917 {
00918     return mOrientation;
00919 }
00920
00921 template<typename T>
00922 void AAX_CParameter<T>::SetNormalizedValue(double normalizedNewValue)
00923 {
00924     T newValue = mTaperDelegate->NormalizedToReal(normalizedNewValue);
00925     this->SetValue(newValue);
00926 }
00927
00928 template<typename T>
00929 double AAX_CParameter<T>::GetNormalizedValue() const
00930 {
00931     T val = GetValue();
00932     return mTaperDelegate->RealToNormalized(val);
00933 }
00934
00935 template<typename T>
00936 bool AAX_CParameter<T>::GetValueString(AAX_CString* valueString) const
00937 {
00938     return mDisplayDelegate->ValueToString(this->GetValue(), valueString);
00939 }
00940
00941 template<typename T>
00942 bool AAX_CParameter<T>::GetValueString(int32_t /*iMaxNumChars*/, AAX_CString* valueString) const
00943 {
00944     return mDisplayDelegate->ValueToString(this->GetValue(), valueString);
00945 }
00946
00947 template<typename T>
00948 bool AAX_CParameter<T>::GetNormalizedValueFromBool(bool /*value*/, double /*normalizedValue*/)
00949 const
00950 {
00951     return false;
00952 }
00953 template<>
00954 bool AAX_CParameter<bool>::GetNormalizedValueFromBool(bool value, double /*normalizedValue*/) const;
00955
00956 template<typename T>
00957 bool AAX_CParameter<T>::GetNormalizedValueFromInt32(int32_t /*value*/, double /*normalizedValue*/) const
00958 {
00959     return false;
00960 }
00961 template<>
00962 bool AAX_CParameter<int32_t>::GetNormalizedValueFromInt32(int32_t value, double /*normalizedValue*/)
00963 const;
00964
00965 template<typename T>
00966 bool AAX_CParameter<T>::GetNormalizedValueFromFloat(float /*value*/, double /*normalizedValue*/)
00967 const
00968 {
00969     return false;
00970 }
00969 template<>
00970 bool AAX_CParameter<float>::GetNormalizedValueFromFloat(float value, double /*normalizedValue*/)
00971 const;
00972
00973 template<typename T>
00974 bool AAX_CParameter<T>::GetNormalizedValueFromDouble(double /*value*/, double /*normalizedValue*/) const
00975 {
00976     return false;
00977 }
00977 template<>
00978 bool AAX_CParameter<double>::GetNormalizedValueFromDouble(double value, double /*normalizedValue*/)
00979 const;

```

```

00979
00980 template <typename T>
00981 bool    AAX_CParameter<T>::GetNormalizedValueFromString(const AAX_CString&    valueString, double
*normalizedValue) const
00982 {
00983     //First, convert the string to a value using the wrapped parameter's display delegate.
00984     T value;
00985     if (!mDisplayDelegate->StringToValue(valueString, &value))
00986         return false;
00987
00988     //Then use the wrapped parameter's taper delegate to convert to a normalized representation.
00989     //If the parameter is out of range, the normalizedValue will be clamped just to be safe.
00990     *normalizedValue = mTaperDelegate->RealToNormalized(value);
00991     return true;
00992 }
00993
00994 template<typename T>
00995 bool    AAX_CParameter<T>::GetBoolFromNormalizedValue(double /*inNormalizedValue*/, bool*
/*value*/) const
00996 {
00997     return false;
00998 }
00999 template <>
01000 bool    AAX_CParameter<bool>::GetBoolFromNormalizedValue(double inNormalizedValue, bool* value)
const;
01001
01002
01003 template<typename T>
01004 bool    AAX_CParameter<T>::GetInt32FromNormalizedValue(double /*inNormalizedValue*/, int32_t*
/*value*/) const
01005 {
01006     return false;
01007 }
01008 template<>
01009 bool    AAX_CParameter<int32_t>::GetInt32FromNormalizedValue(double inNormalizedValue, int32_t*
value) const;
01010
01011 template<typename T>
01012 bool    AAX_CParameter<T>::GetFloatFromNormalizedValue(double /*inNormalizedValue*/, float*
/*value*/) const
01013 {
01014     return false;
01015 }
01016 template<>
01017 bool    AAX_CParameter<float>::GetFloatFromNormalizedValue(double inNormalizedValue, float* value)
const;
01018
01019 template<typename T>
01020 bool    AAX_CParameter<T>::GetDoubleFromNormalizedValue(double /*inNormalizedValue*/, double*
/*value*/) const
01021 {
01022     return false;
01023 }
01024 template<>
01025 bool    AAX_CParameter<double>::GetDoubleFromNormalizedValue(double inNormalizedValue, double*
value) const;
01026
01027 template <typename T>
01028 bool    AAX_CParameter<T>::GetStringFromNormalizedValue(double normalizedValue, AAX_CString&
valueString) const
01029 {
01030     T value = mTaperDelegate->NormalizedToReal(normalizedValue);
01031     if (!mDisplayDelegate->ValueToString(value, &valueString))
01032         return false;
01033
01034     //If the parameter is out of range, we should probably return false, even though we clamped the
normalizedValue already just to be safe.
01035     if ((value > mTaperDelegate->GetMaximumValue()) || (value < mTaperDelegate->GetMinimumValue()))
01036         return false;
01037     return true;
01038 }
01039
01040 template <typename T>
01041 bool    AAX_CParameter<T>::GetStringFromNormalizedValue(double normalizedValue, int32_t iMaxNumChars,
AAX_CString&    valueString) const
01042 {
01043     T value = mTaperDelegate->NormalizedToReal(normalizedValue);
01044     if (!mDisplayDelegate->ValueToString(value, iMaxNumChars, &valueString))
01045         return false;
01046
01047     //If the parameter is out of range, we should probably return false, even though we clamped the
normalizedValue already just to be safe.
01048     if ((value > mTaperDelegate->GetMaximumValue()) || (value < mTaperDelegate->GetMinimumValue()))
01049         return false;
01050     return true;
01051 }
01052

```

```

01053 template<typename T>
01054 bool    AAX_CParameter<T>::SetValueFromString(const AAX_CString&    newValueString)
01055 {
01056     T newValue;
01057     if (!mDisplayDelegate->StringToValue(newValueString, &newValue))
01058         return false;
01059     SetValue(newValue);
01060     return true;
01061 }
01062
01063 template<typename T>
01064 void    AAX_CParameter<T>::SetTaperDelegate(AAX_ITaperDelegateBase& inTaperDelegate, bool
inPreserveValue)
01065 {
01066     double    normalizeValue = this->GetNormalizedValue ();
01067
01068     AAX_ITaperDelegate<T>* oldDelegate = mTaperDelegate;
01069     mTaperDelegate = ((AAX_ITaperDelegate<T> &) inTaperDelegate).Clone();
01070     delete oldDelegate;
01071
01072     mNeedNotify = true;
01073     if ( inPreserveValue )
01074         this->SetValue ( mValue.Get() );
01075     else this->UpdateNormalizedValue ( normalizeValue );
01076 }
01077
01078 template<typename T>
01079 void    AAX_CParameter<T>::SetDisplayDelegate(AAX_IDisplayDelegateBase& inDisplayDelegate)
01080 {
01081     AAX_IDisplayDelegate<T>* oldDelegate = mDisplayDelegate;
01082     mDisplayDelegate = ((AAX_IDisplayDelegate<T> &) inDisplayDelegate).Clone();
01083     delete oldDelegate;
01084
01085     if (mAutomationDelegate != 0)
01086         mAutomationDelegate->PostCurrentValue(this->Identifier(), this->GetNormalizedValue());
01087     //<DMT> Make sure GUIs are all notified of the change.
01088 }
01089
01089 template<typename T>
01090 const AAX_ITaperDelegate<T>*    AAX_CParameter<T>::TaperDelegate() const
01091 {
01092     return mTaperDelegate;
01093 }
01094
01095 template<typename T>
01096 const AAX_IDisplayDelegate<T>*    AAX_CParameter<T>::DisplayDelegate() const
01097 {
01098     return mDisplayDelegate;
01099 }
01100
01101 template<typename T>
01102 bool    AAX_CParameter<T>::Automatable() const
01103 {
01104     return mAutomatable;
01105 }
01106
01107 template<typename T>
01108 void    AAX_CParameter<T>::SetAutomationDelegate ( AAX_IAutomationDelegate * iAutomationDelegate )
01109 {
01110     //Remove the old automation delegate
01111     if ( mAutomationDelegate )
01112     {
01113         mAutomationDelegate->UnregisterParameter ( this->Identifier() );
01114     }
01115
01116     //Add the new automation delegate, wrapped by the versioning layer.
01117     mAutomationDelegate = iAutomationDelegate;
01118     if ( mAutomationDelegate )
01119         mAutomationDelegate->RegisterParameter ( this->Identifier() );
01120 }
01121
01122 template<typename T>
01123 void    AAX_CParameter<T>::Touch()
01124 {
01125     //<DT> Always send the touch command, even if the control isn't automatable.
01126     if (mAutomationDelegate)
01127         mAutomationDelegate->PostTouchRequest( this->Identifier() );
01128 }
01129
01130 template<typename T>
01131 void    AAX_CParameter<T>::Release()
01132 {
01133     //<DT> Always send the release command, even if the control isn't automatable.
01134     if (mAutomationDelegate)
01135         mAutomationDelegate->PostReleaseRequest( this->Identifier() );
01136 }
01137

```



```

01138
01140 #if 0
01141 #pragma mark -
01142 #pragma mark AAX_CStatelessParameter
01143 #endif
01145
01152 class AAX_CStatelessParameter : public AAX_IParameter
01153 {
01154 public:
01155     AAX_CStatelessParameter(AAX_CParamID identifier, const AAX_IString& name, const AAX_IString&
inValueString)
01156     : mNames(name)
01157     , mID(identifier)
01158     , mAutomationDelegate(NULL)
01159     , mValueString(inValueString)
01160     {
01161     }
01162
01163     AAX_CStatelessParameter(const AAX_IString& identifier, const AAX_IString& name, const AAX_IString&
inValueString)
01164     : mNames(name)
01165     , mID(identifier)
01166     , mAutomationDelegate(NULL)
01167     , mValueString(inValueString)
01168     {
01169     }
01170
01171     AAX_DEFAULT_DTOR_OVERRIDE(AAX_CStatelessParameter);
01172
01173     AAX_IParameterValue* CloneValue() const AAX_OVERRIDE { return NULL; }
01174
01175     AAX_CParamID Identifier() const AAX_OVERRIDE { return mID.CString(); }
01176     void SetName(const AAX_CString& name) AAX_OVERRIDE
01177     {
01178         mNames.SetPrimary(name);
01179         if (mAutomationDelegate) {
01180             mAutomationDelegate->ParameterNameChanged(this->Identifier());
01181         }
01182     }
01183
01184     const AAX_CString& Name() const AAX_OVERRIDE { return mNames.Primary(); }
01185     void AddShortenedName(const AAX_CString& name) AAX_OVERRIDE { mNames.Add(name); }
01186     const AAX_CString& ShortenedName(int32_t iNumCharacters) const AAX_OVERRIDE { return
mNames.Get(iNumCharacters); }
01187     void ClearShortenedNames() AAX_OVERRIDE { mNames.Clear(); }
01188
01189     bool Automatable() const AAX_OVERRIDE { return false; }
01190     void SetAutomationDelegate(AAX_IAutomationDelegate* iAutomationDelegate) AAX_OVERRIDE
01191     {
01192         //Remove the old automation delegate
01193         if (mAutomationDelegate)
01194         {
01195             mAutomationDelegate->UnregisterParameter(this->Identifier());
01196         }
01197
01198         //Add the new automation delegate, wrapped by the versioning layer.
01199         mAutomationDelegate = iAutomationDelegate;
01200         if (mAutomationDelegate)
01201             mAutomationDelegate->RegisterParameter(this->Identifier());
01202     }
01203
01204     void Touch() AAX_OVERRIDE { if (mAutomationDelegate) mAutomationDelegate->PostTouchRequest(
this->Identifier()); }
01205     void Release() AAX_OVERRIDE { if (mAutomationDelegate)
mAutomationDelegate->PostReleaseRequest(this->Identifier()); }
01206
01207     void SetNormalizedValue(double /*newNormalizedValue*/) AAX_OVERRIDE {}
01208     double GetNormalizedValue() const AAX_OVERRIDE { return 0.; }
01209     void SetNormalizedDefaultValue(double /*normalizedDefault*/) AAX_OVERRIDE {}
01210     double GetNormalizedDefaultValue() const AAX_OVERRIDE { return 0.; }
01211     void SetToDefaultValue() AAX_OVERRIDE {}
01212     void SetNumberOfSteps(uint32_t /*numSteps*/) AAX_OVERRIDE {}
01213     uint32_t GetNumberOfSteps() const AAX_OVERRIDE { return 1; }
01214     uint32_t GetStepValue() const AAX_OVERRIDE { return 0; }
01215     double GetNormalizedValueFromStep(uint32_t /*iStep*/) const AAX_OVERRIDE { return 0.; }
01216     uint32_t GetStepValueFromNormalizedValue(double /*normalizedValue*/) const AAX_OVERRIDE {
return 0; }
01217     void SetStepValue(uint32_t /*iStep*/) AAX_OVERRIDE {}
01218
01219     bool GetValueString(AAX_CString* valueString) const AAX_OVERRIDE { if (valueString)
*valueString = mValueString; return true; }
01220     bool GetValueString(int32_t /*iMaxNumChars*/, AAX_CString* valueString) const AAX_OVERRIDE
{ return this->GetValueString(valueString); }
01221     bool GetNormalizedValueFromBool(bool /*value*/, double* normalizedValue) const AAX_OVERRIDE
{ if (normalizedValue) { *normalizedValue = 0.; } return true; }
01222     bool GetNormalizedValueFromInt32(int32_t /*value*/, double* normalizedValue) const
AAX_OVERRIDE { if (normalizedValue) { *normalizedValue = 0.; } return true; }
01223     bool GetNormalizedValueFromFloat(float /*value*/, double* normalizedValue) const
AAX_OVERRIDE { if (normalizedValue) { *normalizedValue = 0.; } return true; }

```

```

01243     bool        GetNormalizedValueFromDouble(double /*value*/, double* normalizedValue) const
AAX_OVERRIDE { if (normalizedValue) { *normalizedValue = 0.; } return true; }
01244     bool        GetNormalizedValueFromString(const AAX_CString& /*valueString*/, double*
normalizedValue) const AAX_OVERRIDE { if (normalizedValue) { *normalizedValue = 0.; } return true; }
01245     bool        GetBoolFromNormalizedValue(double /*normalizedValue*/, bool* value) const AAX_OVERRIDE
{ if (value) { *value = false; } return true; }
01246     bool        GetInt32FromNormalizedValue(double /*normalizedValue*/, int32_t* /*value*/) const
AAX_OVERRIDE { return false; }
01247     bool        GetFloatFromNormalizedValue(double /*normalizedValue*/, float* /*value*/) const
AAX_OVERRIDE { return false; }
01248     bool        GetDoubleFromNormalizedValue(double /*normalizedValue*/, double* /*value*/) const
AAX_OVERRIDE { return false; }
01249     bool        GetStringFromNormalizedValue(double /*normalizedValue*/, AAX_CString& valueString)
const AAX_OVERRIDE { valueString = mValueString; return true; }
01250     bool        GetStringFromNormalizedValue(double normalizedValue, int32_t /*iMaxNumChars*/,
AAX_CString& valueString) const AAX_OVERRIDE { return
this->GetStringFromNormalizedValue(normalizedValue, valueString); }
01251     bool        SetValueFromString(const AAX_CString& newValueString) AAX_OVERRIDE { mValueString =
newValueString; return true; }
01253
01258     bool        GetValueAsBool(bool* value) const AAX_OVERRIDE { if (value) { *value = false; } return
true; }
01259     bool        GetValueAsInt32(int32_t* /*value*/) const AAX_OVERRIDE { return false; }
01260     bool        GetValueAsFloat(float* /*value*/) const AAX_OVERRIDE { return false; }
01261     bool        GetValueAsDouble(double* /*value*/) const AAX_OVERRIDE { return false; }
01262     bool        GetValueAsString(AAX_IString* /*value*/) const AAX_OVERRIDE { return false; }
01263     bool        SetValueWithBool(bool /*value*/) AAX_OVERRIDE { return true; }
01264     bool        SetValueWithInt32(int32_t /*value*/) AAX_OVERRIDE { return false; }
01265     bool        SetValueWithFloat(float /*value*/) AAX_OVERRIDE { return false; }
01266     bool        SetValueWithDouble(double /*value*/) AAX_OVERRIDE { return false; }
01267     bool        SetValueWithString(const AAX_IString& value) AAX_OVERRIDE { mValueString = value;
return true; }
01269
01270     void        SetType( AAX_EParameterType /*iControlType*/ ) AAX_OVERRIDE {};
01271     AAX_EParameterType    GetType() const AAX_OVERRIDE { return AAX_eParameterType_Discrete; }
01272
01273     void        SetOrientation( AAX_EParameterOrientation /*iOrientation*/ ) AAX_OVERRIDE {}
01274     AAX_EParameterOrientation    GetOrientation() const AAX_OVERRIDE { return
AAX_eParameterOrientation_Default; }
01275
01276     void SetTaperDelegate ( AAX_ITaperDelegateBase & /*inTaperDelegate*/, bool /*inPreserveValue*/ )
AAX_OVERRIDE {};
01277     void SetDisplayDelegate ( AAX_IDisplayDelegateBase & /*inDisplayDelegate*/ ) AAX_OVERRIDE {};
01278
01283     void        UpdateNormalizedValue(double /*newNormalizedValue*/) AAX_OVERRIDE {};
01285
01286 protected:
01287     AAX_CStringAbbreviations mNames;
01288     AAX_CString mID;
01289     AAX_IAutomationDelegate * mAutomationDelegate;
01290     AAX_CString mValueString;
01291 };
01292
01293
01294
01295
01296 #endif //AAX_CParameter_H

```

15.99 AAX_CParameterManager.h File Reference

```

#include "AAX_CParameter.h"
#include "AAX.h"
#include <vector>
#include <map>

```

15.99.1 Description

A container object for plug-in parameters.

Classes

- class [AAX_CParameterManager](#)
A container object for plug-in parameters.

15.100 AAX_CParameterManager.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CPARAMETERMANAGER_H
00023 #define AAX_CPARAMETERMANAGER_H
00024
00025 #include "AAX_CParameter.h"
00026 #include "AAX.h"
00027
00028 #include <vector>
00029 #include <map>
00030
00031
00032
00033
00034 class AAX_IAutomationDelegate;
00035
00048 class AAX_CParameterManager
00049 {
00050 public:
00051     AAX_CParameterManager();
00052     ~AAX_CParameterManager();
00053
00063     void Initialize(AAX_IAutomationDelegate* iAutomationDelegateUnknown);
00064
00069     int32_t NumParameters() const;
00070
00079     void RemoveParameterByID(AAX_CParamID identifier);
00080
00086     void RemoveAllParameters();
00087
00094     AAX_IParameter* GetParameterByID(AAX_CParamID identifier);
00095
00102     const AAX_IParameter* GetParameterByID(AAX_CParamID identifier) const;
00103
00112     AAX_IParameter* GetParameterByName(const char* name);
00113
00122     const AAX_IParameter* GetParameterByName(const char* name) const;
00123
00133     AAX_IParameter* GetParameter(int32_t index);
00134
00144     const AAX_IParameter* GetParameter(int32_t index) const;
00145
00151     int32_t GetParameterIndex(AAX_CParamID identifier) const;
00152
00160     void AddParameter(AAX_IParameter* param);
00161
00169     void RemoveParameter(AAX_IParameter* param);
00170
00171 protected:
00172
00173     AAX_IAutomationDelegate* mAutomationDelegate; //This object is not ref-counted
00174     here. Do not delete it. It is ref counted by this object's parent.
00175     std::vector<AAX_IParameter*> mParameters;
00176     std::map<std::string, AAX_IParameter*> mParametersMap;
00177 };
00178
00179
00180
00181 #endif // AAX_CPARAMETERMANAGER_H

```

15.101 AAX_CPercentDisplayDelegateDecorator.h File Reference

```

#include "AAX_IDisplayDelegateDecorator.h"
#include <cmath>

```

15.101.1 Description

A percent display delegate decorator.

Classes

- class [AAX_CPercentDisplayDelegateDecorator< T >](#)
A percent decorator conforming to [AAX_IDisplayDelegateDecorator](#).

Macros

- #define [AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H](#)

15.101.2 Macro Definition Documentation

15.101.2.1 AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H

```
#define AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H
```

15.102 AAX_CPercentDisplayDelegateDecorator.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #pragma once
00023
00024 #ifndef AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H
00025 #define AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H
00026
00027 #include "AAX_IDisplayDelegateDecorator.h"
00028
00029 #include <cmath>
00030
00031
00054 template <typename T>
00055 class AAX_CPercentDisplayDelegateDecorator : public AAX_IDisplayDelegateDecorator<T>
00056 {
00057 public:
00058     AAX_CPercentDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>& displayDelegate);
00059
00060     //Virtual Overrides
```

```

00061     AAX_CPercentDisplayDelegateDecorator<T>* Clone() const AAX_OVERRIDE;
00062     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00063     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const
AAX_OVERRIDE;
00064     bool StringToValue(const AAX_CString& valueString, T* value) const AAX_OVERRIDE;
00065 };
00066
00067 template <typename T>
00068 AAX_CPercentDisplayDelegateDecorator<T>::AAX_CPercentDisplayDelegateDecorator(const
AAX_IDisplayDelegate<T>& displayDelegate) :
00069     AAX_IDisplayDelegateDecorator<T>(displayDelegate)
00070 {
00071 }
00072
00073 template <typename T>
00074 AAX_CPercentDisplayDelegateDecorator<T>* AAX_CPercentDisplayDelegateDecorator<T>::Clone() const
00075 {
00076     return new AAX_CPercentDisplayDelegateDecorator(*this);
00077 }
00078
00079 template <typename T>
00080 bool AAX_CPercentDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString) const
00081 {
00082     value *= 100;
00083     bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00084     *valueString += AAX_CString("%");
00085     return succeeded;
00086 }
00087
00088 template <typename T>
00089 bool AAX_CPercentDisplayDelegateDecorator<T>::ValueToString(T value, int32_t maxNumChars, AAX_CString*
valueString) const
00090 {
00091     value *= 100;
00092     bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars-1,
valueString); //<DMT> Make room for percentage symbol.
00093     *valueString += AAX_CString("%");
00094     return succeeded;
00095 }
00096
00097
00098 template <typename T>
00099 bool AAX_CPercentDisplayDelegateDecorator<T>::StringToValue(const AAX_CString& valueString, T* value)
const
00100 {
00101     //Just call through if there is obviously no unit string.
00102     if (valueString.Length() <= 2)
00103     {
00104         bool success = AAX_IDisplayDelegateDecorator<T>::StringToValue(valueString, value);
00105         *value /= 100.0f;
00106         return success;
00107     }
00108
00109     //Just call through if the end of this string does not match the unit string.
00110     AAX_CString unitSubString;
00111     valueString.SubString(valueString.Length() - 1, 1, &unitSubString);
00112     if (unitSubString != AAX_CString("%"))
00113     {
00114         bool success = AAX_IDisplayDelegateDecorator<T>::StringToValue(valueString, value);
00115         *value /= 100.0f;
00116         return success;
00117     }
00118
00119     //Call through with the stripped down value string.
00120     AAX_CString valueSubString;
00121     valueString.SubString(0, valueString.Length() - 1, &valueSubString);
00122     bool success = AAX_IDisplayDelegateDecorator<T>::StringToValue(valueSubString, value);
00123     *value /= 100.0f;
00124     return success;
00125 }
00126
00127
00128 #endif
00129

```

15.103 AAX_CPieceWiseLinearTaperDelegate.h File Reference

```

#include "AAX_ITaperDelegate.h"
#include "AAX.h"
#include <cmath>

```

15.103.1 Description

A piece-wise linear taper delegate.

Classes

- class [AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >](#)

A piece-wise linear taper conforming to [AAX_ITaperDelegate](#).

15.104 AAX_CPieceWiseLinearTaperDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00013
00014 #ifndef AAX_CPIECEWISELINEARTAPERDELEGATE_H
00015 #define AAX_CPIECEWISELINEARTAPERDELEGATE_H
00016
00017 #include "AAX_ITaperDelegate.h"
00018 #include "AAX.h" //for types
00019
00020 #include <cmath> //for floor()
00021
00022
00023 template <typename T, int32_t RealPrecision=100>
00024 class AAX_CPieceWiseLinearTaperDelegate : public AAX_ITaperDelegate<T>
00025 {
00026 public:
00027     AAX_CPieceWiseLinearTaperDelegate(const double* normalizedValues, const T* realValues, int32_t
numValues);
00028
00029     AAX_CPieceWiseLinearTaperDelegate(const AAX_CPieceWiseLinearTaperDelegate& other); //Explicit
copy constructor because there are internal arrays.
00030     ~AAX_CPieceWiseLinearTaperDelegate();
00031
00032     //Virtual AAX_ITaperDelegate Overrides
00033     AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>* Clone() const AAX_OVERRIDE;
00034     T GetMinimumValue() const AAX_OVERRIDE { return mMinValue; }
00035     T GetMaximumValue() const AAX_OVERRIDE { return mMaxValue; }
00036     T ConstrainRealValue(T value) const AAX_OVERRIDE;
00037     T NormalizedToReal(double normalizedValue) const AAX_OVERRIDE;
00038     double RealToNormalized(T realValue) const AAX_OVERRIDE;
00039
00040 protected:
00041     T Round(double iValue) const;
00042
00043 private:
00044     double* mNormalizedValues;
00045     T* mRealValues;
00046     int32_t mNumValues;
00047     T mMinValue; //Really just an optimization
00048     T mMaxValue; //Really just an optimization
00049 };
00050
00051 template <typename T, int32_t RealPrecision>
00052 AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::Round(double iValue) const
00053 {
00054     if (RealPrecision > 0)
00055         return static_cast<T>(floor(iValue * RealPrecision + 0.5) / RealPrecision);
00056     else
00057         return static_cast<T>(iValue);
00058 }

```

```

00098
00099 template <typename T, int32_t RealPrecision>
00100 AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::AAX_CPieceWiseLinearTaperDelegate(const double*
normalizedValues, const T* realValues, int32_t numValues) : AAX_ITaperDelegate<T>(),
00101     mNormalizedValues(0),
00102     mRealValues(0),
00103     mNumValues(0),
00104     mMinValue(0),
00105     mMaxValue(0)
00106 {
00107     mNormalizedValues = new double[numValues];
00108     mRealValues = new T[numValues];
00109     mNumValues = numValues;
00110
00111     if (numValues > 0)
00112     {
00113         mMaxValue = realValues[0];
00114         mMinValue = realValues[0];
00115     }
00116     for (int32_t i=0; i< numValues; i++)
00117     {
00118         mNormalizedValues[i] = normalizedValues[i];
00119         mRealValues[i] = realValues[i];
00120         if (mRealValues[i] > mMaxValue)
00121             mMaxValue = mRealValues[i];
00122         if (mRealValues[i] < mMinValue)
00123             mMinValue = mRealValues[i];
00124     }
00125 }
00126
00127 template <typename T, int32_t RealPrecision>
00128 AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::AAX_CPieceWiseLinearTaperDelegate(const
AAX_CPieceWiseLinearTaperDelegate& other) : AAX_ITaperDelegate<T>(),
00129     mNormalizedValues(0),
00130     mRealValues(0),
00131     mNumValues(0),
00132     mMinValue(0),
00133     mMaxValue(0)
00134 {
00135     mNormalizedValues = new double[other.mNumValues];
00136     mRealValues = new T[other.mNumValues];
00137     mNumValues = other.mNumValues;
00138     mMaxValue = other.mMaxValue;
00139     mMinValue = other.mMinValue;
00140     for (int32_t i=0; i< mNumValues; i++)
00141     {
00142         mNormalizedValues[i] = other.mNormalizedValues[i];
00143         mRealValues[i] = other.mRealValues[i];
00144     }
00145 }
00146
00147 template <typename T, int32_t RealPrecision>
00148 AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::~AAX_CPieceWiseLinearTaperDelegate()
00149 {
00150     mNumValues = 0;
00151     delete [] mNormalizedValues;
00152     delete [] mRealValues;
00153 }
00154
00155
00156 template <typename T, int32_t RealPrecision>
00157 AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>*
AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::Clone() const
00158 {
00159     return new AAX_CPieceWiseLinearTaperDelegate(*this);
00160 }
00161
00162 template <typename T, int32_t RealPrecision>
00163 T AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::ConstrainRealValue(T value) const
00164 {
00165     if (mMinValue == mMaxValue)
00166         return mMinValue;
00167
00168     if (RealPrecision)
00169         value = Round(value); //reduce the precision to get proper rounding behavior with
integers.
00170
00171     const T& highValue = mMaxValue > mMinValue ? mMaxValue : mMinValue;
00172     const T& lowValue = mMaxValue > mMinValue ? mMinValue : mMaxValue;
00173
00174     if (value > highValue)
00175         return highValue;
00176     if (value < lowValue)
00177         return lowValue;
00178
00179     return value;
00180 }

```

```

00181
00182 template <typename T, int32_t RealPrecision>
00183 T      AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::NormalizedToReal(double normalizedValue)
00184     const
00185 {
00186     // Clip to normalized range.
00187     if (normalizedValue > 1.0)
00188         normalizedValue = 1.0;
00189     if (normalizedValue < 0.0)
00190         normalizedValue = 0.0;
00191
00192     // This is basically linear interpolation so let's first find the bounding normalized points from
00193     our specified array.
00194     int32_t mLowerIndex = 0;
00195     int32_t mUpperIndex = 0;
00196     for (int32_t i=1; i<mNumValues; i++)
00197     {
00198         mUpperIndex++;
00199         if (mNormalizedValues[i] >= normalizedValue)
00200             break;
00201         mLowerIndex++;
00202     }
00203     // Do the interpolation.
00204     double delta = normalizedValue - mNormalizedValues[mLowerIndex];
00205     double slope = double(mRealValues[mUpperIndex] - mRealValues[mLowerIndex]) /
00206     (mNormalizedValues[mUpperIndex] - mNormalizedValues[mLowerIndex]);
00207     double interpolatedValue = mRealValues[mLowerIndex] + (delta * slope);
00208
00209     return ConstrainRealValue(static_cast<T>(interpolatedValue));
00210 }
00211
00212 template <typename T, int32_t RealPrecision>
00213 double AAX_CPieceWiseLinearTaperDelegate<T, RealPrecision>::RealToNormalized(T realValue) const
00214 {
00215     realValue = ConstrainRealValue(realValue);
00216
00217     // This is basically linear interpolation so let's first find the bounding normalized points from
00218     our specified array.
00219     int32_t mLowerIndex = 0;
00220     int32_t mUpperIndex = 0;
00221     if (mRealValues[0] < mRealValues[mNumValues-1])
00222     {
00223         //Increasing real values (positive slope)
00224         for (int32_t i=1; i<mNumValues; i++)
00225         {
00226             mUpperIndex++;
00227             if (mRealValues[i] >= realValue)
00228                 break;
00229             mLowerIndex++;
00230         }
00231     }
00232     else
00233     {
00234         //Decreasing real values (negative slope)
00235         for (int32_t i=1; i<mNumValues; i++)
00236         {
00237             mUpperIndex++;
00238             if (mRealValues[i] <= realValue)
00239                 break;
00240             mLowerIndex++;
00241         }
00242     }
00243     // Do the interpolation.
00244     double delta = realValue - mRealValues[mLowerIndex];
00245     double slope = (mRealValues[mUpperIndex] == mRealValues[mLowerIndex]) ? 0.5 :
00246     double(mNormalizedValues[mUpperIndex] - mNormalizedValues[mLowerIndex]) / (mRealValues[mUpperIndex] -
00247     mRealValues[mLowerIndex]);
00248     double interpolatedValue = mNormalizedValues[mLowerIndex] + (delta * slope);
00249
00250     return static_cast<T>(interpolatedValue);
00251 }
00252 #endif //AAX_CPIECEWISELINEARTAPERDELEGATE_H

```


15.105 AAX_CRangeTaperDelegate.h File Reference

```
#include "AAX_ITaperDelegate.h"
#include "AAX.h"
#include <cmath>
#include <vector>
```

15.105.1 Description

A range taper delegate decorator.

Classes

- class [AAX_CRangeTaperDelegate< T, RealPrecision >](#)
A piecewise-linear taper conforming to [AAX_ITaperDelegate](#).

15.106 AAX_CRangeTaperDelegate.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CRANGETAPERDELEGATE_H
00023 #define AAX_CRANGETAPERDELEGATE_H
00024
00025 #include "AAX_ITaperDelegate.h"
00026 #include "AAX.h" //for types
00027
00028 #include <cmath> //for floor()
00029 #include <vector>
00030
00031
00079 template <typename T, int32_t RealPrecision=1000>
00080 class AAX_CRangeTaperDelegate : public AAX_ITaperDelegate<T>
00081 {
00082 public:
00096 AAX_CRangeTaperDelegate(T* range, double* rangesSteps, long numRanges, bool useSmartRounding =
    true);
00097 AAX_CRangeTaperDelegate( const AAX_CRangeTaperDelegate& rhs);
00098 AAX_CRangeTaperDelegate& operator=( AAX_CRangeTaperDelegate& rhs );
00099
00100 //Virtual Overrides
00101 AAX_CRangeTaperDelegate<T, RealPrecision>* Clone() const AAX_OVERRIDE;
00102 T GetMinimumValue() const AAX_OVERRIDE { return mMinValue; }
00103 T GetMaximumValue() const AAX_OVERRIDE { return mMaxValue; }
00104 T ConstrainRealValue(T value) const AAX_OVERRIDE;
00105 T NormalizedToReal(double normalizedValue) const AAX_OVERRIDE;
00106 double RealToNormalized(T realValue) const AAX_OVERRIDE;
00107
00108 protected:
00109 T Round(double iValue) const;
00110 T SmartRound(double value) const;
00111
00112 private:
```

```

00113     T          mMinValue;
00114     T          mMaxValue;
00115     long       mNumRanges;
00116     std::vector<T> mRanges;
00117     std::vector<double> mRangesSteps;
00118     std::vector<double> mRangesPercents;
00119     std::vector<double> mRangesStepsCount;
00120     bool       mUseSmartRounding;
00121 };
00122
00123 template <typename T, int32_t RealPrecision>
00124 AAX_CRangeTaperDelegate<T, RealPrecision>::AAX_CRangeTaperDelegate(T* ranges, double* rangesSteps,
00125     long numRanges, bool useSmartRounding) :
00126     AAX_ITaperDelegate<T>(),
00127     mMinValue(*ranges),
00128     mMaxValue(*(ranges + numRanges)),
00129     mNumRanges(numRanges),
00130     mRanges( ranges, ranges + numRanges + 1),
00131     mRangesSteps( rangesSteps, rangesSteps + numRanges),
00132     mUseSmartRounding( useSmartRounding )
00133 {
00134     mRangesStepsCount.reserve(numRanges);
00135     mRangesPercents.reserve(numRanges);
00136     int i = 0;
00137     for (; i < mNumRanges; i++)
00138     {
00139         mRangesStepsCount.push_back( (mRanges.at(i + 1) - mRanges.at(i)) / mRangesSteps.at(i));
00140     }
00141     double numSteps = 0;
00142     for (i = 0; i < mNumRanges; i++)
00143     {
00144         numSteps += mRangesStepsCount.at(i);
00145     }
00146     for (i = 0; i < mNumRanges; i++)
00147     {
00148         mRangesPercents.push_back( mRangesStepsCount.at(i) / numSteps );
00149     }
00150 }
00151
00152 template <typename T, int32_t RealPrecision>
00153 AAX_CRangeTaperDelegate<T, RealPrecision>::AAX_CRangeTaperDelegate( const
00154     AAX_CRangeTaperDelegate<T, RealPrecision>& rhs) :
00155     mMinValue(rhs.mMinValue),
00156     mMaxValue(rhs.mMaxValue),
00157     mNumRanges(rhs.mNumRanges),
00158     mRanges( rhs.mRanges.begin(), rhs.mRanges.end()),
00159     mRangesSteps( rhs.mRangesSteps.begin(), rhs.mRangesSteps.end()),
00160     mRangesPercents( rhs.mRangesPercents.begin(), rhs.mRangesPercents.end()),
00161     mRangesStepsCount( rhs.mRangesStepsCount.begin(), rhs.mRangesStepsCount.end()),
00162     mUseSmartRounding( rhs.mUseSmartRounding )
00163 {
00164 }
00165
00166 template <typename T, int32_t RealPrecision>
00167 AAX_CRangeTaperDelegate<T, RealPrecision>& AAX_CRangeTaperDelegate<T, RealPrecision>::operator=(
00168     AAX_CRangeTaperDelegate<T, RealPrecision>& rhs)
00169 {
00170     if (this == &rhs)
00171         return *this;
00172     this->mMinValue = rhs.mMinValue;
00173     this->mMaxValue = rhs.mMaxValue;
00174     this->mNumRanges = rhs.mNumRanges;
00175     this->mRanges.assign( rhs.mRanges.begin(), rhs.mRanges.end());
00176     this->mRangesSteps.assign( rhs.mRangesSteps.begin(), rhs.mRangesSteps.end());
00177     this->mRangesPercents.assign( rhs.mRangesPercents.begin(), rhs.mRangesPercents.end());
00178     this->mRangesStepsCount.assign( rhs.mRangesStepsCount.begin(), rhs.mRangesStepsCount.end());
00179     return *this;
00180 }
00181
00182 template <typename T, int32_t RealPrecision>
00183 T AAX_CRangeTaperDelegate<T, RealPrecision>::Round(double iValue) const
00184 {
00185     return ((0 >= RealPrecision) ? static_cast<T>(iValue) :
00186         (0 <= iValue) ? floor( iValue*RealPrecision + 0.5f ) / RealPrecision :
00187         ceil( iValue*RealPrecision - 0.5f ) / RealPrecision
00188     );
00189 }
00190
00191 template <typename T, int32_t RealPrecision>
00192 AAX_CRangeTaperDelegate<T, RealPrecision>* AAX_CRangeTaperDelegate<T, RealPrecision>::Clone()
00193     const
00194 {
00195     return new AAX_CRangeTaperDelegate<T, RealPrecision>(*this);
00196 }

```

```

00196 template <typename T, int32_t RealPrecision>
00197 T AAX_CRangeTaperDelegate<T, RealPrecision>::ConstrainRealValue(T value) const
00198 {
00199     if (mMinValue == mMaxValue)
00200         return mMinValue;
00201
00202     if (RealPrecision)
00203         value = Round(value); //reduce the precision to get proper rounding behavior with
                                integers.
00204
00205     const T& highValue = mMaxValue > mMinValue ? mMaxValue : mMinValue;
00206     const T& lowValue = mMaxValue > mMinValue ? mMinValue : mMaxValue;
00207
00208     if (value > highValue)
00209         return highValue;
00210     if (value < lowValue)
00211         return lowValue;
00212
00213     return value;
00214 }
00215
00216 template <typename T, int32_t RealPrecision>
00217 T AAX_CRangeTaperDelegate<T, RealPrecision>::NormalizedToReal(double normalizedValue) const
00218 {
00219     double percentTotal = normalizedValue;
00220
00221     double percent = 0.0;
00222     long i = 0;
00223     for (; i < mNumRanges; i++)
00224     {
00225         if ((percentTotal >= percent) && (percentTotal < (percent + mRangesPercents.at(i))))
00226             break;
00227         percent += mRangesPercents.at(i);
00228     }
00229
00230     double extValue;
00231     if (i == mNumRanges)
00232         extValue = mMaxValue; // our control is 100% of maximum
00233     else
00234         extValue = mRanges.at(i) + ((mRanges.at(i+1) - mRanges.at(i))*(percentTotal - percent)) /
                                (mRangesPercents.at(i));
00235
00236     T realValue = T(extValue);
00237     if (mUseSmartRounding)
00238         realValue = SmartRound(extValue); //reduce the precision to get proper rounding behavior
                                with integers.
00239
00240     return ConstrainRealValue(realValue);
00241 }
00242
00243 template <typename T, int32_t RealPrecision>
00244 double AAX_CRangeTaperDelegate<T, RealPrecision>::RealToNormalized(T realValue) const
00245 {
00246     realValue = ConstrainRealValue(realValue);
00247
00248     double percentTotal = 0.0;
00249     long i = 0;
00250     for (; i < mNumRanges; i++)
00251     {
00252         if ((realValue >= mRanges[i]) && (realValue < mRanges[i+1]))
00253             break;
00254         percentTotal += mRangesPercents[i];
00255     }
00256
00257     if (i == mNumRanges)
00258         percentTotal = 1.0; // our control is 100% of maximum
00259     else if (mRanges.at(i + 1) == mRanges.at(i))
00260         ; // no action; total percent does not change
00261     else
00262         percentTotal += (realValue - mRanges.at(i))/(mRanges.at(i + 1) - mRanges.at(i)) *
                                mRangesPercents.at(i);
00263
00264     double normalizedValue = percentTotal;
00265     return normalizedValue;
00266 }
00267
00268 template <typename T, int32_t RealPrecision>
00269 T AAX_CRangeTaperDelegate<T, RealPrecision>::SmartRound(double value) const
00270 {
00271     int32_t i = 0;
00272     for (; i < mNumRanges; i++)
00273     {
00274         if ((value >= mRanges.at(i)) && (value < mRanges.at(i + 1)))
00275             break;
00276         if (i == mNumRanges - 1)
00277             break;
00278     }

```

```

00279
00280     int32_t longVal = 0;
00281     if (value >= 0)
00282         longVal = int32_t(floor(value / mRangesSteps.at(i) + 0.5));
00283     else
00284         longVal = int32_t(ceil(value / mRangesSteps.at(i) - 0.5));
00285
00286     return static_cast<double>(longVal) * mRangesSteps.at(i);
00287 }
00288
00289
00290 #endif

```

15.107 AAX_CSessionDocumentClient.h File Reference

```

#include "AAX_ISessionDocumentClient.h"
#include <memory>

```

Classes

- class [AAX_CSessionDocumentClient](#)
Default implementation of the [AAX_ISessionDocumentClient](#) interface.

Macros

- #define [AAX_CSessionDocumentClient_H](#)

15.107.1 Macro Definition Documentation

15.107.1.1 AAX_CSessionDocumentClient_H

```
#define AAX_CSessionDocumentClient_H
```

15.108 AAX_CSessionDocumentClient.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019 #ifndef AAX_CSessionDocumentClient_H
00020 #define AAX_CSessionDocumentClient_H

```

```

00021
00022 #include "AAX_I_SessionDocumentClient.h"
00023 #include <memory>
00024
00025 #ifdef __clang__
00026 #pragma clang diagnostic push
00027 #pragma clang diagnostic ignored "-Wunused-parameter"
00028 #endif
00029
00030 class AAX_I_Controller;
00031 class AAX_I_EffectParameters;
00032 class AAX_I_SessionDocument;
00033 class AAX_V_SessionDocument;
00034
00035
00038 class AAX_C_SessionDocumentClient : public AAX_I_SessionDocumentClient
00039 {
00040 public:
00041
00042     AAX_C_SessionDocumentClient(void);
00043     ~AAX_C_SessionDocumentClient(void) AAX_OVERRIDE;
00044
00045 public:
00046
00053     AAX_Result Initialize (IACFUnknown * iUnknown) AAX_OVERRIDE;
00057     AAX_Result Uninitialize (void) AAX_OVERRIDE;
00059
00066     AAX_Result SetSessionDocument (IACFUnknown * iSessionDocument) AAX_OVERRIDE;
00068
00075     AAX_Result NotificationReceived(/* AAX_ENotificationEvent */ AAX_CTypeID /*inNotificationType*/,
const void * /*inNotificationData*/, uint32_t /*inNotificationDataSize*/) AAX_OVERRIDE { return
AAX_SUCCESS; }
00077
00078 protected:
00079
00091     virtual AAX_Result SessionDocumentWillChange() { return AAX_SUCCESS; }
00100     virtual AAX_Result SessionDocumentChanged() { return AAX_SUCCESS; }
00102
00110     AAX_I_Controller* GetController (void);
00111     const AAX_I_Controller* GetController (void) const;
00112
00117     AAX_I_EffectParameters* GetEffectParameters (void);
00118     const AAX_I_EffectParameters* GetEffectParameters (void) const;
00119
00124     std::shared_ptr<AAX_I_SessionDocument> GetSessionDocument (void);
00125     std::shared_ptr<const AAX_I_SessionDocument> GetSessionDocument (void) const;
00127
00128 private:
00129     void ClearInternalState();
00130
00131     //These are private, but they all have protected accessors.
00132     AAX_UNIQUE_PTR(AAX_I_Controller) mController;
00133     AAX_I_EffectParameters * mEffectParameters;
00134     std::shared_ptr<AAX_V_SessionDocument> mSessionDocument;
00135 };
00136
00137 #ifdef __clang__
00138 #pragma clang diagnostic pop
00139 #endif
00140
00141 #endif // AAX_C_SessionDocumentClient

```

15.109 AAX_CStateDisplayDelegate.h File Reference

```

#include "AAX_IDisplayDelegate.h"
#include "AAX_CString.h"
#include <vector>

```

15.109.1 Description

A state display delegate.

Classes

- class [AAX_CStateDisplayDelegate< T >](#)

A generic display format conforming to [AAX_IDisplayDelegate](#).

15.110 AAX_CStateDisplayDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CSTATEDISPLAYDELEGATE_H
00023 #define AAX_CSTATEDISPLAYDELEGATE_H
00024
00025 #include "AAX_IDisplayDelegate.h"
00026 #include "AAX_CString.h"
00027
00028 #include <vector>
00029 #if defined(WINDOWS_VERSION) || defined(LINUX_VERSION)
00030 #include <algorithm>
00031 #endif
00032
00033
00034
00044 template <typename T>
00045 class AAX_CStateDisplayDelegate : public AAX_IDisplayDelegate<T>
00046 {
00047 public:
00055     explicit AAX_CStateDisplayDelegate( const char * iStateStrings[], T iMinState = 0 );
00056
00065     explicit AAX_CStateDisplayDelegate( int32_t inNumStates, const char * iStateStrings[], T iMinState
= 0 );
00066
00072     explicit AAX_CStateDisplayDelegate( const std::vector<AAX_IString*>& iStateStrings, T iMinState =
0 );
00073
00074     AAX_CStateDisplayDelegate(const AAX_CStateDisplayDelegate& other);
00075
00076     //Virtual Overrides
00077     AAX_IDisplayDelegate<T>* Clone() const AAX_OVERRIDE;
00078     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00079     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString)
const AAX_OVERRIDE;
00080     bool StringToValue(const AAX_CString& valueString, T* value) const
AAX_OVERRIDE;
00081
00082     //AAX_CStateDisplayDelegate
00083     void AddShortenedStrings( const char * iStateStrings[], int iLength );
00084     bool Compare( const AAX_CString& valueString, const AAX_CString&
stateString ) const;
00085
00086 private:
00087     AAX_CStateDisplayDelegate(); //private constructor to prevent its use externally.
00088
00089     T mMinState;
00090     std::vector<AAX_CString> mStateStrings;
00091
00092     struct StringTable
00093     {
00094         int mStrLength;
00095         std::vector<AAX_CString> mStateStrings;
00096     };
00097     static bool StringTableSortFunc(struct StringTable i, struct StringTable j)
00098     {
00099         return (i.mStrLength < j.mStrLength);
00100     }
00101

```

```

00102     std::vector<struct StringTable> mShortenedStrings;
00103 };
00104
00105 template <typename T>
00106 AAX_CStateDisplayDelegate<T>::AAX_CStateDisplayDelegate( const char * iStateStrings[], T iMinState /*
    = 0 */ )
00107 {
00108     mMinState = iMinState;
00109     for ( int index = 0; iStateStrings[ index ] != 0; ++index )
00110         mStateStrings.push_back( AAX_CString( iStateStrings[ index ] ) );
00111 }
00112
00113 template <typename T>
00114 AAX_CStateDisplayDelegate<T>::AAX_CStateDisplayDelegate( int32_t inNumStates, const char *
    iStateStrings[], T iMinState /* = 0 */ )
00115 {
00116     mMinState = iMinState;
00117     for ( int index = 0; (index < inNumStates) && (iStateStrings[ index ] != 0); ++index )
00118         mStateStrings.push_back( AAX_CString( iStateStrings[ index ] ) );
00119 }
00120
00121 template <typename T>
00122 AAX_CStateDisplayDelegate<T>::AAX_CStateDisplayDelegate( const std::vector<AAX_IString*>&
    iStateStrings, T iMinState /* = 0 */ )
00123 {
00124     mMinState = iMinState;
00125     for ( std::vector<AAX_IString*>::const_iterator iter = iStateStrings.begin(); iter !=
    iStateStrings.end(); ++iter )
00126     {
00127         if (*iter)
00128         {
00129             mStateStrings.push_back( *(*iter) );
00130         }
00131     }
00132 }
00133
00134 template <typename T>
00135 AAX_CStateDisplayDelegate<T>::AAX_CStateDisplayDelegate( const AAX_CStateDisplayDelegate & iOther )
00136 {
00137     mMinState = iOther.mMinState;
00138
00139     std::vector<AAX_CString*>::const_iterator iter = iOther.mStateStrings.begin();
00140     for ( ; iter != iOther.mStateStrings.end(); ++iter )
00141         mStateStrings.push_back( AAX_CString( *iter ) );
00142
00143     if ( iOther.mShortenedStrings.size() > 0 )
00144     {
00145         for ( int i = 0; i < (int)iOther.mShortenedStrings.size(); i++ )
00146             mShortenedStrings.push_back( iOther.mShortenedStrings.at(i) );
00147     }
00148 }
00149
00150 template <typename T>
00151 void AAX_CStateDisplayDelegate<T>::AddShortenedStrings( const char * iStateStrings[], int iStrLength )
00152 {
00153     struct StringTable shortendTable;
00154     shortendTable.mStrLength = iStrLength;
00155     for ( int index = 0; iStateStrings[ index ] != 0; ++index )
00156         shortendTable.mStateStrings.push_back( AAX_CString( iStateStrings[ index ] ) );
00157     mShortenedStrings.push_back(shortendTable);
00158
00159     // keep structure sorted by str lengths
00160     std::sort(mShortenedStrings.begin(), mShortenedStrings.end(),
    AAX_CStateDisplayDelegate::StringTableSortFunc );
00161 }
00162
00163 template <typename T>
00164 AAX_IDisplayDelegate<T>* AAX_CStateDisplayDelegate<T>::Clone() const
00165 {
00166     return new AAX_CStateDisplayDelegate(*this);
00167 }
00168
00169 template <typename T>
00170 bool AAX_CStateDisplayDelegate<T>::ValueToString(T value, AAX_CString* valueString) const
00171 {
00172     T index = value - mMinState;
00173     if ( index >= (T) 0 && index < (T) mStateStrings.size() )
00174     {
00175         *valueString = mStateStrings[ index ];
00176         return true;
00177     }
00178     return false;
00179 }
00180
00181
00182 template <typename T>
00183 bool AAX_CStateDisplayDelegate<T>::ValueToString(T value, int32_t maxNumChars, AAX_CString*

```

```

    valueString) const
00184 {
00185     // if we don't have any shortened strings, just return the full length version
00186     if ( mShortenedStrings.size() == 0 )
00187         return this->ValueToString(value, valueString);
00188
00189     // iterate through shortened strings from longest to shortest
00190     // taking the first set that is short enough
00191     T index = value - mMinState;
00192
00193     if ( index < (T) 0 || index >= (T) mStateStrings.size() )
00194         return true;
00195
00196     // first see if the normal string is short enough
00197     if ( mStateStrings[ index ].Length() < uint32_t(maxNumChars) )
00198     {
00199         *valueString = mStateStrings[ index ];
00200         return true;
00201     }
00202
00203     for ( int i = (int)mShortenedStrings.size()-1; i >= 0; i-- )
00204     {
00205         struct StringTable shortStrings = mShortenedStrings.at(i);
00206         if ( shortStrings.mStrLength <= maxNumChars )
00207         {
00208             if ( index >= (T) 0 && index < (T) shortStrings.mStateStrings.size() )
00209             {
00210                 *valueString = shortStrings.mStateStrings[ index ];
00211                 return true;
00212             }
00213         }
00214     }
00215
00216     // if we can't find one short enough, just use the shortest version we can find
00217     struct StringTable shortestStrings = mShortenedStrings.at(0);
00218     if ( index >= (T) 0 && index < (T) shortestStrings.mStateStrings.size() )
00219     {
00220         *valueString = shortestStrings.mStateStrings[ index ];
00221         return true;
00222     }
00223
00224     return false;
00225 }
00226
00227 template <typename T>
00228 bool    AAX_CStateDisplayDelegate<T>::StringToValue(const AAX_CString& valueString, T* value) const
00229 {
00230     std::vector<AAX_CString>::const_iterator iter = mStateStrings.begin();
00231     for ( T index = 0; iter != mStateStrings.end(); ++index, ++iter )
00232     {
00233         if (Compare(valueString,*iter))
00234         {
00235             *value = index + mMinState;
00236             return true;
00237         }
00238     }
00239
00240     *value = mMinState;
00241     return false;
00242 }
00243
00244 template <typename T>
00245 bool    AAX_CStateDisplayDelegate<T>::Compare( const AAX_CString& valueString, const AAX_CString&
00246     stateString ) const
00247 {
00248     return valueString==stateString;
00249 }
00250
00251
00252
00253
00254 #endif //AAX_CSTATEDISPLAYDELEGATE_H

```

15.111 AAX_CStateTaperDelegate.h File Reference

```

#include "AAX_ITaperDelegate.h"
#include "AAX.h"
#include <cmath>

```


15.111.1 Description

A state taper delegate (similar to a linear taper delegate.)

Classes

- class [AAX_CStateTaperDelegate< T >](#)
A linear taper conforming to [AAX_ITaperDelegate](#).

15.112 AAX_CStateTaperDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CSTATETAPERDELEGATE_H
00023 #define AAX_CSTATETAPERDELEGATE_H
00024
00025 #include "AAX_ITaperDelegate.h"
00026 #include "AAX.h" //for types
00027
00028 #include <cmath> //for floor()
00029
00030
00042 template <typename T>
00043 class AAX_CStateTaperDelegate : public AAX_ITaperDelegate<T>
00044 {
00045 public:
00053     AAX_CStateTaperDelegate(T minValue=0, T maxValue=1);
00054
00055     //Virtual Overrides
00056     AAX_CStateTaperDelegate<T>* Clone() const AAX_OVERRIDE;
00057     T GetMinimumValue() const AAX_OVERRIDE { return mMinValue; }
00058     T GetMaximumValue() const AAX_OVERRIDE { return mMaxValue; }
00059     T ConstrainRealValue(T value) const AAX_OVERRIDE;
00060     T NormalizedToReal(double normalizedValue) const AAX_OVERRIDE;
00061     double RealToNormalized(T realValue) const AAX_OVERRIDE;
00062
00063 private:
00064     T mMinValue;
00065     T mMaxValue;
00066 };
00067
00068 template <typename T>
00069 AAX_CStateTaperDelegate<T>::AAX_CStateTaperDelegate(T minValue, T maxValue) :
00070     AAX_ITaperDelegate<T>(),
00071     mMinValue(minValue),
00072     mMaxValue(maxValue)
00073 {
00074 }
00075
00076 template <typename T>
00077 AAX_CStateTaperDelegate<T>* AAX_CStateTaperDelegate<T>::Clone() const
00078 {
00079     return new AAX_CStateTaperDelegate(*this);
00080 }
00081
00082 template <typename T>
00083 T AAX_CStateTaperDelegate<T>::ConstrainRealValue(T value) const
00084 {
00085     if (mMinValue == mMaxValue)

```

```

00086         return mMinValue;
00087
00088     const T& highValue = mMaxValue > mMinValue ? mMaxValue : mMinValue;
00089     const T& lowValue = mMaxValue > mMinValue ? mMinValue : mMaxValue;
00090
00091     if (value > highValue)
00092         return highValue;
00093     if (value < lowValue)
00094         return lowValue;
00095
00096     return value;
00097 }
00098
00099 template <typename T>
00100 T      AAX_CStateTaperDelegate<T>::NormalizedToReal(double normalizedValue) const
00101 {
00102     double doubleRealValue = normalizedValue * (double(mMaxValue) - double(mMinValue)) +
double(mMinValue);
00103     if ( doubleRealValue >= 0 )
00104         doubleRealValue += 0.5;
00105     else doubleRealValue -= 0.5;
00106     return ConstrainRealValue(static_cast<T>(doubleRealValue));
00107 }
00108
00109 template <typename T>
00110 double AAX_CStateTaperDelegate<T>::RealToNormalized(T realValue) const
00111 {
00112     realValue = ConstrainRealValue(realValue);
00113     double normalizedValue = (mMaxValue == mMinValue) ? 0.5 : (double(realValue) - double(mMinValue))
/ (double(mMaxValue) - double(mMinValue));
00114     return normalizedValue;
00115 }
00116
00117
00118
00119
00120 #endif //AAX_CSTATETAPERDELEGATE_H

```

15.113 AAX_CString.h File Reference

```

#include "AAX_IString.h"
#include "AAX.h"
#include <string>
#include <map>

```

15.113.1 Description

A generic AAX string class with similar functionality to `std::string`.

Classes

- class [AAX_CString](#)
A generic AAX string class with similar functionality to `std::string`
- class [AAX_CStringAbbreviations](#)
Helper class to store a collection of name abbreviations.

Macros

- #define [AAX_CSTRING_H](#)

Functions

- [AAX_CString operator+](#) ([AAX_CString](#) lhs, const [AAX_CString](#) &rhs)
- [AAX_CString operator+](#) ([AAX_CString](#) lhs, const char *rhs)
- [AAX_CString operator+](#) (const char *lhs, const [AAX_CString](#) &rhs)

15.113.2 Macro Definition Documentation

15.113.2.1 AAX_CSTRING_H

```
#define AAX_CSTRING_H
```

15.113.3 Function Documentation

15.113.3.1 operator+() [1/3]

```
AAX\_CString operator+ (  
    AAX\_CString lhs,  
    const AAX\_CString & rhs ) [inline]
```

15.113.3.2 operator+() [2/3]

```
AAX\_CString operator+ (  
    AAX\_CString lhs,  
    const char * rhs ) [inline]
```

15.113.3.3 operator+() [3/3]

```
AAX\_CString operator+ (  
    const char * lhs,  
    const AAX\_CString & rhs ) [inline]
```

15.114 AAX_CString.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2017, 2021, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012
00019  /*=====*/
00020
00021  #pragma once
00022
00023  #ifndef AAX_CSTRING_H
00024  #define AAX_CSTRING_H
00025
00026
00027  #include "AAX_IString.h"
00028  #include "AAX.h"
00029
00030  #include <string>
00031  #include <map>
00032
00033
00035  #if 0
00036  #pragma mark -
00037  #endif
00039
00043  class AAX_CString : public AAX_IString
00044  {
00045  public:
00046      static const uint32_t kInvalidIndex = static_cast<uint32_t>(-1);
00047      static const uint32_t kMaxStringLength = static_cast<uint32_t>(-2);
00048
00049      // AAX_IString Virtual Overrides
00050      uint32_t Length() const AAX_OVERRIDE;
00051      uint32_t MaxLength() const AAX_OVERRIDE;
00052      const char * Get () const AAX_OVERRIDE;
00053      void Set ( const char * iString ) AAX_OVERRIDE;
00054      AAX_IString & operator=(const AAX_IString & iOther) AAX_OVERRIDE;
00055      AAX_IString & operator=(const char * iString) AAX_OVERRIDE;
00056
00058      AAX_CString();
00059
00061      AAX_CString(const char* str);
00062
00064      explicit AAX_CString(const std::string& str);
00065
00067      AAX_CString(const AAX_CString& other);
00068
00070      AAX_CString(const AAX_IString& other);
00071
00073      AAX_DEFAULT_MOVE_CTOR(AAX_CString);
00074
00075
00077      std::string& StdString();
00078
00080      const std::string& StdString() const;
00081
00083      AAX_CString& operator=(const AAX_CString& other);
00084
00086      AAX_CString & operator=(const std::string& other);
00087
00089      AAX_CString & operator=(AAX_CString&& other);
00090
00092      friend std::ostream& operator<< (std::ostream& os, const AAX_CString& str);
00093
00095      friend std::istream& operator>> (std::istream& os, AAX_CString& str);
00096
00097
00098      // String Formatting Functions
00099      void Clear();
00100      bool Empty() const;
00101      AAX_CString& Erase(uint32_t pos, uint32_t n);
00102      AAX_CString& Append(const AAX_CString& str);
00103      AAX_CString& Append(const char* str);
00104      AAX_CString& AppendNumber(double number, int32_t precision);
00105      AAX_CString& AppendNumber(int32_t number);
00106      AAX_CString& AppendHex(int32_t number, int32_t width);

```

```

00107     AAX_CString& Insert(uint32_t pos, const AAX_CString& str);
00108     AAX_CString& Insert(uint32_t pos, const char* str);
00109     AAX_CString& InsertNumber(uint32_t pos, double number, int32_t precision);
00110     AAX_CString& InsertNumber(uint32_t pos, int32_t number);
00111     AAX_CString& InsertHex(uint32_t pos, int32_t number, int32_t width);
00112     AAX_CString& Replace(uint32_t pos, uint32_t n, const AAX_CString& str);
00113     AAX_CString& Replace(uint32_t pos, uint32_t n, const char* str);
00114     uint32_t FindFirst(const AAX_CString& findStr) const;
00115     uint32_t FindFirst(const char* findStr) const;
00116     uint32_t FindFirst(char findChar) const;
00117     uint32_t FindLast(const AAX_CString& findStr) const;
00118     uint32_t FindLast(const char* findStr) const;
00119     uint32_t FindLast(char findChar) const;
00120     const char* CString() const;
00121     bool ToDouble(double* oValue) const;
00122     bool ToInteger(int32_t* oValue) const;
00123     void SubString(uint32_t pos, uint32_t n, AAX_IString* outputStr) const;
00124     bool Equals(const AAX_CString& other) const { return operator==(other); }
00125     bool Equals(const char* other) const { return operator==(other); }
00126     bool Equals(const std::string& other) const { return operator==(other); } //beware of
STL variations between binaries.
00127
00128     // Operator Overrides
00129     bool operator==(const AAX_CString& other) const;
00130     bool operator==(const char* otherStr) const;
00131     bool operator==(const std::string& otherStr) const; //beware of STL variations
between binaries.
00132     bool operator!=(const AAX_CString& other) const;
00133     bool operator!=(const char* otherStr) const;
00134     bool operator!=(const std::string& otherStr) const; //beware of STL variations
between binaries.
00135     bool operator<(const AAX_CString& other) const;
00136     bool operator>(const AAX_CString& other) const;
00137     const char& operator[](uint32_t index) const;
00138     char& operator[](uint32_t index);
00139     AAX_CString& operator+=(const AAX_CString& str);
00140     AAX_CString& operator+=(const std::string& str);
00141     AAX_CString& operator+=(const char* str);
00142
00143 protected:
00144     std::string mString;
00145 };
00146
00147 // Non-member operators
00148 inline AAX_CString operator+(AAX_CString lhs, const AAX_CString& rhs)
00149 {
00150     lhs += rhs;
00151     return lhs;
00152 }
00153 inline AAX_CString operator+(AAX_CString lhs, const char* rhs)
00154 {
00155     lhs += rhs;
00156     return lhs;
00157 }
00158 inline AAX_CString operator+(const char* lhs, const AAX_CString& rhs)
00159 {
00160     return AAX_CString(lhs) + rhs;
00161 }
00162
00163
00164 #if 0
00165 #pragma mark -
00166 #endif
00167
00168 class AAX_CStringAbbreviations
00169 {
00170 public:
00171     explicit AAX_CStringAbbreviations(const AAX_CString& inPrimary)
00172     : mPrimary(inPrimary)
00173     , mAbbreviations()
00174     {
00175     }
00176
00177     void SetPrimary(const AAX_CString& inPrimary) { mPrimary = inPrimary; }
00178     const AAX_CString& Primary() const { return mPrimary; }
00179
00180     void Add(const AAX_CString& inAbbreviation)
00181     {
00182         uint32_t stringLength = inAbbreviation.Length();
00183         mAbbreviations[stringLength] = inAbbreviation; //Does a string copy into the map.
00184     }
00185
00186     const AAX_CString& Get(int32_t inNumCharacters) const
00187     {
00188         //More characters than the primary string or no specific shortened names.
00189         if ((inNumCharacters >= int32_t(mPrimary.Length())) || (mAbbreviations.empty()) || (0 >
inNumCharacters))

```

```

00194         return mPrimary;
00195
00196         std::map<uint32_t, AAX_CString>::const_iterator iter =
mAbbreviations.upper_bound(static_cast<uint32_t>(inNumCharacters));
00197
00198         //If the iterator is already pointing to shortest string, return that.
00199         if (iter == mAbbreviations.begin())
00200             return iter->second;
00201
00202         //lower_bound() will return the iterator that is larger than the desired value, so decrement
the iterator.
00203         --iter;
00204         return iter->second;
00205     }
00206
00207     void Clear() { mAbbreviations.clear(); }
00208
00209 private:
00210     AAX_CString mPrimary;
00211     std::map<uint32_t, AAX_CString> mAbbreviations;
00212 };
00213
00214 #endif //AAX_CSTRING_H

```

15.115 AAX_CStringDataBuffer.h File Reference

```

#include "AAX_IDataBuffer.h"
#include "AAX.h"
#include <string>
#include <limits>
#include <type_traits>

```

Classes

- class [AAX_CStringDataBufferOfType< T >](#)
A convenience class for string data buffers.
- class [AAX_CStringDataBuffer](#)

Macros

- #define [AAX_CStringDataBuffer_H](#)

15.115.1 Macro Definition Documentation

15.115.1.1 AAX_CStringDataBuffer_H

```
#define AAX_CStringDataBuffer_H
```

15.116 AAX_CStringDataBuffer.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019
00020 #ifndef AAX_CStringDataBuffer_H
00021 #define AAX_CStringDataBuffer_H
00022
00023 #include "AAX_IDataBuffer.h"
00024 #include "AAX.h"
00025
00026 #include <string>
00027 #include <limits>
00028 #include <type_traits>
00029
00030
00036 template <AAX_CTypeID T>
00037 class AAX_CStringDataBufferOfType : public AAX_IDataBuffer
00038 {
00039 public:
00040     explicit AAX_CStringDataBufferOfType (std::string const & inData) : mData{inData} {}
00041     explicit AAX_CStringDataBufferOfType (std::string && inData) : mData{inData} {}
00042     explicit AAX_CStringDataBufferOfType (const char * inData) : mData{inData ? std::string{inData} :
std::string{}} {}
00043
00044     AAX_CStringDataBufferOfType(AAX_CStringDataBufferOfType const &) = delete;
00045     AAX_CStringDataBufferOfType(AAX_CStringDataBufferOfType &&) = delete;
00046
00047     ~AAX_CStringDataBufferOfType (void) AAX_OVERRIDE = default;
00048
00049     AAX_CStringDataBufferOfType& operator= (AAX_CStringDataBufferOfType const & other) = delete;
00050     AAX_CStringDataBufferOfType& operator= (AAX_CStringDataBufferOfType && other) = delete;
00051
00052     AAX_Result Type(AAX_CTypeID * oType) const AAX_OVERRIDE {
00053         if (!oType) { return AAX_ERROR_NULL_ARGUMENT; }
00054         *oType = T;
00055         return AAX_SUCCESS;
00056     }
00057     AAX_Result Size(int32_t * oSize) const AAX_OVERRIDE {
00058         if (!oSize) { return AAX_ERROR_NULL_ARGUMENT; }
00059         auto const size = mData.size() + 1; // null termination
00060         static_assert(std::numeric_limits<decltype(size)>::max() >=
std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max(),
00061             "size variable may not represent all positive values of oSize");
00062         if (size > std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max()) {
00063             return AAX_ERROR_SIGNED_INT_OVERFLOW;
00064         }
00065         *oSize = static_cast<std::remove_pointer<decltype(oSize)>::type>(size);
00066         return AAX_SUCCESS;
00067     }
00068     AAX_Result Data(void const ** oBuffer) const AAX_OVERRIDE {
00069         if (!oBuffer) { return AAX_ERROR_NULL_ARGUMENT; }
00070         *oBuffer = mData.c_str();
00071         return AAX_SUCCESS;
00072     }
00073 private:
00074     std::string mData;
00075 };
00076
00080 class AAX_CStringDataBuffer : public AAX_IDataBuffer
00081 {
00082 public:
00083     AAX_CStringDataBuffer (AAX_CTypeID inType, std::string const & inData) : mType{inType},
mData{inData} {}
00084     AAX_CStringDataBuffer (AAX_CTypeID inType, std::string && inData) : mType{inType}, mData{inData}
{}
00085     AAX_CStringDataBuffer (AAX_CTypeID inType, const char * inData) : mType{inType}, mData{inData ?
std::string{inData} : std::string{}} {}
00086
00087     AAX_CStringDataBuffer(AAX_CStringDataBuffer const &) = delete;
00088     AAX_CStringDataBuffer(AAX_CStringDataBuffer &&) = delete;

```

```

00089
00090 ~AAX_CStringDataBuffer (void) AAX_OVERRIDE = default;
00091
00092 AAX_CStringDataBuffer& operator= (AAX_CStringDataBuffer const & other) = delete;
00093 AAX_CStringDataBuffer& operator= (AAX_CStringDataBuffer && other) = delete;
00094
00095 AAX_Result Type(AAX_CTypeID * oType) const AAX_OVERRIDE {
00096     if (!oType) { return AAX_ERROR_NULL_ARGUMENT; }
00097     *oType = mType;
00098     return AAX_SUCCESS;
00099 }
00100 AAX_Result Size(int32_t * oSize) const AAX_OVERRIDE {
00101     if (!oSize) { return AAX_ERROR_NULL_ARGUMENT; }
00102     auto const size = mData.size() + 1; // null termination
00103     static_assert(std::numeric_limits<decltype(size)>::max() >=
std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max(),
00104         "size variable may not represent all positive values of oSize");
00105     if (size > std::numeric_limits<std::remove_pointer<decltype(oSize)>::type>::max()) {
00106         return AAX_ERROR_SIGNED_INT_OVERFLOW;
00107     }
00108     *oSize = static_cast<std::remove_pointer<decltype(oSize)>::type>(size);
00109     return AAX_SUCCESS;
00110 }
00111 AAX_Result Data(void const ** oBuffer) const AAX_OVERRIDE {
00112     if (!oBuffer) { return AAX_ERROR_NULL_ARGUMENT; }
00113     *oBuffer = mData.c_str();
00114     return AAX_SUCCESS;
00115 }
00116 private:
00117     AAX_CTypeID const mType;
00118     std::string mData;
00119 };
00120
00121 #endif

```

15.117 AAX_CStringDisplayDelegate.h File Reference

```

#include "AAX_IDisplayDelegate.h"
#include <sstream>
#include <map>

```

15.117.1 Description

A string display delegate.

Classes

- class [AAX_CStringDisplayDelegate< T >](#)
A string, or list, display format conforming to [AAX_IDisplayDelegate](#).

15.118 AAX_CStringDisplayDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *

```



```

00011  */
00012
00019  /*=====*/
00020
00021
00022 #ifndef AAX_CSTRINGDISPLAYDELEGATE_H
00023 #define AAX_CSTRINGDISPLAYDELEGATE_H
00024
00025 #include "AAX_IDisplayDelegate.h"
00026 #include <sstream>
00027 #include <map>
00028
00029
00042 template <typename T>
00043 class AAX_CStringDisplayDelegate : public AAX_IDisplayDelegate<T>
00044 {
00045 public:
00057     AAX_CStringDisplayDelegate(const std::map<T,AAX_CString>& stringMap);
00058
00059     //Virtual Overrides
00060     AAX_IDisplayDelegate<T>* Clone() const AAX_OVERRIDE;
00061     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00062     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const
00063         AAX_OVERRIDE;
00064     bool StringToValue(const AAX_CString& valueString, T* value) const AAX_OVERRIDE;
00065 protected:
00066     std::map<T, AAX_CString> mStringMap;
00067     std::map<AAX_CString, T> mInverseStringMap;
00068 };
00069
00070
00071
00072 template <typename T>
00073 AAX_CStringDisplayDelegate<T>::AAX_CStringDisplayDelegate(const std::map<T,AAX_CString>& stringMap) :
00074     AAX_IDisplayDelegate<T>(),
00075     mStringMap(stringMap),
00076     mInverseStringMap()
00077 {
00078     //Construct an inverse string map from our already copied internal copy of the string map.
00079     //This inverse map is used for stringToValue conversion.
00080     typename std::map<T,AAX_CString>::iterator valueStringIterator = mStringMap.begin();
00081     while ( valueStringIterator != mStringMap.end() )
00082     {
00083         mInverseStringMap.insert(std::pair<AAX_CString, T>(valueStringIterator->second,
00084             valueStringIterator->first));
00085         valueStringIterator++;
00086     }
00087
00088     template <typename T>
00089     AAX_CStringDisplayDelegate<T>* AAX_CStringDisplayDelegate<T>::Clone() const
00090     {
00091         return new AAX_CStringDisplayDelegate(*this);
00092     }
00093
00094     template <typename T>
00095     bool AAX_CStringDisplayDelegate<T>::ValueToString(T value, AAX_CString* valueString) const
00096     {
00097         typename std::map<T,AAX_CString>::const_iterator mapPairIterator = mStringMap.find(value);
00098         if( mapPairIterator != mStringMap.end() )
00099         {
00100             *valueString = mapPairIterator->second;
00101             return true;
00102         }
00103         *valueString = AAX_CString("String Not Found");
00104         return false;
00105     }
00106
00107     template <typename T>
00108     bool AAX_CStringDisplayDelegate<T>::ValueToString(T value, int32_t /*maxNumChars*/,
00109         AAX_CString* valueString) const
00110     {
00111         // First, get the full length string.
00112         bool result = this->ValueToString(value, valueString);
00113         //<DMT> TODO: Shorten the string based on the number of characters...
00114         return result;
00115     }
00116
00117     template <typename T>
00118     bool AAX_CStringDisplayDelegate<T>::StringToValue(const AAX_CString& valueString, T* value)
00119         const
00120     {
00121         typename std::map<AAX_CString, T>::const_iterator mapPairIterator =
00122             mInverseStringMap.find(valueString);

```

```

00122     if( mapPairIterator != mInverseStringMap.end() )
00123     {
00124         *value = mapPairIterator->second;
00125         return true;
00126     }
00127     *value = 0;
00128     return false;
00129 }
00130
00131
00132
00133
00134 #endif //AAX_CSTRINGDISPLAYDELEGATE_H

```

15.119 AAX_CTaskAgent.h File Reference

```

#include "AAX_ITaskAgent.h"
#include <memory>

```

15.119.1 Description

A default implementation of the [AAX_ITaskAgent](#) interface.

Classes

- class [AAX_CTaskAgent](#)
Default implementation of the [AAX_ITaskAgent](#) interface.

15.120 AAX_CTaskAgent.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CTaskAgent_H
00023 #define AAX_CTaskAgent_H
00024
00025 #include "AAX_ITaskAgent.h"
00026 #include <memory>
00027
00028 class AAX_IController;
00029 class AAX_IEffectParameters;
00030 class AAX_ITask;
00031
00041 class AAX_CTaskAgent : public AAX_ITaskAgent
00042 {
00043 public:
00044     AAX_CTaskAgent (void) = default;
00045     ~AAX_CTaskAgent (void) AAX_OVERRIDE;
00046
00047 public:

```

```

00048
00052     AAX_Result Initialize (IACFUnknown * iController ) AAX_OVERRIDE;
00053     AAX_Result Uninitialize (void) AAX_OVERRIDE;
00055
00067     AAX_Result AddTask(IACFUnknown * iTask) AAX_OVERRIDE;
00068     AAX_Result CancelAllTasks() AAX_OVERRIDE;
00070
00071 protected:
00072
00078     virtual AAX_Result AddTask(std::unique_ptr<AAX_ITask> iTask);
00079
00083     virtual AAX_Result ReceiveTask(std::unique_ptr<AAX_ITask> iTask);
00084
00085 public:
00086
00093     AAX_IController* GetController (void) { return mController; };
00097     AAX_IEffectParameters* GetEffectParameters (void) { return mEffectParameters; }
00099
00100 private:
00101     void ReleaseObjects();
00102
00103     AAX_IController* mController = nullptr;
00104     AAX_IEffectParameters* mEffectParameters = nullptr;
00105 };
00106
00107
00108 #endif

```

15.121 AAX_CUnitDisplayDelegateDecorator.h File Reference

```
#include "AAX_IDisplayDelegateDecorator.h"
```

15.121.1 Description

A unit display delgate decorator.

Classes

- class [AAX_CUnitDisplayDelegateDecorator< T >](#)
A unit type decorator conforming to [AAX_IDisplayDelegateDecorator](#).

15.122 AAX_CUnitDisplayDelegateDecorator.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CUNITDISPLAYDELEGATEDECORATOR_H
00023 #define AAX_CUNITDISPLAYDELEGATEDECORATOR_H
00024
00025 #include "AAX_IDisplayDelegateDecorator.h"
00026

```

```

00027
00042 template <typename T>
00043 class AAX_CUnitDisplayDelegateDecorator : public AAX_IDisplayDelegateDecorator<T>
00044 {
00045 public:
00054     AAX_CUnitDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>& displayDelegate, const
AAX_CString& unitString);
00055
00056     //Virtual Overrides
00057     AAX_CUnitDisplayDelegateDecorator<T>* Clone() const AAX_OVERRIDE;
00058     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00059     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const
AAX_OVERRIDE;
00060     bool StringToValue(const AAX_CString& valueString, T* value) const AAX_OVERRIDE;
00061
00062 protected:
00063     const AAX_CString mUnitString;
00064 };
00065
00066
00067
00068
00069 template <typename T>
00070 AAX_CUnitDisplayDelegateDecorator<T>::AAX_CUnitDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>&
displayDelegate, const AAX_CString& unitString) :
00071     AAX_IDisplayDelegateDecorator<T>(displayDelegate),
00072     mUnitString(unitString)
00073 {
00074
00075 }
00076
00077 template <typename T>
00078 AAX_CUnitDisplayDelegateDecorator<T>* AAX_CUnitDisplayDelegateDecorator<T>::Clone() const
00079 {
00080     return new AAX_CUnitDisplayDelegateDecorator(*this);
00081 }
00082
00083 template <typename T>
00084 bool AAX_CUnitDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString)
const
00085 {
00086     bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00087     *valueString += mUnitString;
00088     return succeeded;
00089 }
00090
00091 template <typename T>
00092 bool AAX_CUnitDisplayDelegateDecorator<T>::ValueToString(T value, int32_t maxNumChars,
AAX_CString* valueString) const
00093 {
00094     bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars, valueString);
00095     uint32_t strlen = valueString->Length();
00096     const uint32_t maxNumCharsUnsigned = (0 <= maxNumChars) ? static_cast<uint32_t>(maxNumChars) : 0;
00097     if (maxNumCharsUnsigned > strlen && (maxNumCharsUnsigned-strlen >= mUnitString.Length()))
00098         *valueString += mUnitString;
00099     return succeeded;
00100 }
00101
00102
00103 template <typename T>
00104 bool AAX_CUnitDisplayDelegateDecorator<T>::StringToValue(const AAX_CString& valueString, T*
value) const
00105 {
00106     //Just call through if there is obviously no unit string.
00107     if (valueString.Length() <= mUnitString.Length())
00108         return AAX_IDisplayDelegateDecorator<T>::StringToValue(valueString, value);
00109
00110     //Just call through if the end of this string does not match the unit string.
00111     AAX_CString unitSubString;
00112     valueString.SubString(valueString.Length() - mUnitString.Length(), mUnitString.Length(),
&unitSubString);
00113     if (unitSubString != mUnitString)
00114         return AAX_IDisplayDelegateDecorator<T>::StringToValue(valueString, value);
00115
00116     //Call through with the stripped down value string.
00117     AAX_CString valueSubString;
00118     valueString.SubString(0, valueString.Length() - mUnitString.Length(), &valueSubString);
00119     return AAX_IDisplayDelegateDecorator<T>::StringToValue(valueSubString, value);
00120 }
00121
00122
00123
00124
00125
00126 #endif //AAX_CUNITDISPLAYDELEGATEDECORATOR_H

```

15.123 AAX_CUnitPrefixDisplayDelegateDecorator.h File Reference

```
#include "AAX_IDisplayDelegateDecorator.h"
```

15.123.1 Description

A unit prefix display delegate decorator.

Classes

- class [AAX_CUnitPrefixDisplayDelegateDecorator< T >](#)
A unit prefix decorator conforming to [AAX_IDisplayDelegateDecorator](#).

15.124 AAX_CUnitPrefixDisplayDelegateDecorator.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_CUNITPREFIXDISPLAYDELEGATEDECORATOR_H
00023 #define AAX_CUNITPREFIXDISPLAYDELEGATEDECORATOR_H
00024
00025 #include "AAX_IDisplayDelegateDecorator.h"
00026
00027
00058 template <typename T>
00059 class AAX_CUnitPrefixDisplayDelegateDecorator : public AAX_IDisplayDelegateDecorator<T>
00060 {
00061 public:
00062     AAX_CUnitPrefixDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>& displayDelegate);
00063
00064     //Virtual overrides
00065     AAX_CUnitPrefixDisplayDelegateDecorator<T>* Clone() const AAX_OVERRIDE;
00066     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00067     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const
00068         AAX_OVERRIDE;
00068     bool StringToValue(const AAX_CString& valueString, T* value) const AAX_OVERRIDE;
00069 };
00070
00071
00072
00073 template <typename T>
00074 AAX_CUnitPrefixDisplayDelegateDecorator<T>::AAX_CUnitPrefixDisplayDelegateDecorator(const
00075     AAX_IDisplayDelegate<T>& displayDelegate) :
00076     AAX_IDisplayDelegateDecorator<T>(displayDelegate)
00077 {
00078 }
00079
00080
00081 template <typename T>
00082 AAX_CUnitPrefixDisplayDelegateDecorator<T>* AAX_CUnitPrefixDisplayDelegateDecorator<T>::Clone()
00083     const
00084 {
00084     return new AAX_CUnitPrefixDisplayDelegateDecorator(*this);
00085 }
```

```

00086
00087 template <typename T>
00088 bool AAX_CUnitPrefixDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString*
valueString) const
00089 {
00090     //Find the proper unit prefix.
00091     T absValue = fabsf(float(value)); //If you fail to compile on this line, you're trying to use
this class with an integer type, which is not supported.
00092     if (absValue >= 1000000.0)
00093     {
00094         value = value / ((T) 1000000.0);
00095         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00096         *valueString += AAX_CString("M");
00097         return succeeded;
00098     }
00099     if (absValue >= ((T) 1000.0))
00100     {
00101         value = value / ((T) 1000.0);
00102         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00103         *valueString += AAX_CString("k");
00104         return succeeded;
00105     }
00106     if (absValue >= ((T) 1.0))
00107     {
00108         return AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00109     }
00110     if (absValue >= ((T) 0.001))
00111     {
00112         value = value / ((T) 0.001);
00113         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00114         *valueString += AAX_CString("m");
00115         return succeeded;
00116     }
00117     if (absValue >= ((T) 0.000001))
00118     {
00119         value = value / ((T) 0.000001);
00120         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00121         *valueString += AAX_CString("u");
00122         return succeeded;
00123     }
00124     return AAX_IDisplayDelegateDecorator<T>::ValueToString(value, valueString);
00125 }
00126
00127 template <typename T>
00128 bool AAX_CUnitPrefixDisplayDelegateDecorator<T>::ValueToString(T value, int32_t maxNumChars,
AAX_CString* valueString) const
00129 {
00130     //Find the proper unit prefix.
00131     //<DMT> The maxNumChars is decremented by 1 in case of the unit modifier being required as this is
more important than precision.
00132
00133     T absValue = fabsf(float(value)); //If you fail to compile on this line, you're trying to use
this class with an integer type, which is not supported.
00134     if (absValue >= 1000000.0)
00135     {
00136         value = value / ((T) 1000000.0);
00137         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars-1,
valueString);
00138         *valueString += AAX_CString("M");
00139         return succeeded;
00140     }
00141     if (absValue >= ((T) 1000.0))
00142     {
00143         value = value / ((T) 1000.0);
00144         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars-1,
valueString);
00145         *valueString += AAX_CString("k");
00146         return succeeded;
00147     }
00148     if (absValue >= ((T) 1.0))
00149     {
00150         return AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars, valueString);
00151     }
00152     if (absValue >= ((T) 0.001))
00153     {
00154         value = value / ((T) 0.001);
00155         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars-1,
valueString);
00156         *valueString += AAX_CString("m");
00157         return succeeded;
00158     }
00159     if (absValue >= ((T) 0.000001))
00160     {
00161         value = value / ((T) 0.000001);
00162         bool succeeded = AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars-1,
valueString);
00163         *valueString += AAX_CString("u");

```

```

00164         return succeeded;
00165     }
00166     return AAX_IDisplayDelegateDecorator<T>::ValueToString(value, maxNumChars, valueString);
00167 }
00168
00169
00170 template <typename T>
00171 bool AAX_CUnitPrefixDisplayDelegateDecorator<T>::StringToValue(const AAX_CString& valueString,
00172     T* value) const
00173 {
00174     //Just call through if there is obviously no unit string.
00175     if (valueString.Length() <= 1)
00176         return AAX_IDisplayDelegateDecorator<T>::StringToValue(valueString, value);
00177
00178     //Just call through if the end of this string does not match the unit string.
00179     AAX_CString valueStringCopy(valueString);
00180     T valueScalar = 1;
00181     T valueDivScalar = 1;
00182     switch(valueString[valueString.Length()-1])
00183     {
00184     case 'M':
00185         valueScalar = ((T) 1000000.0);
00186         valueStringCopy.Erase(valueString.Length()-1, 1);
00187         break;
00188     case 'k':
00189         valueScalar = ((T) 1000.0);
00190         valueStringCopy.Erase(valueString.Length()-1, 1);
00191         break;
00192     case 'm':
00193         valueScalar = ((T) 0.001);
00194         valueStringCopy.Erase(valueString.Length()-1, 1);
00195         break;
00196     case 'u':
00197         // Rounding errors occur when trying to use 0.000001 so went to a div scalar instead.
00198         // See bug https://audio-jira.avid.com/browse/PTSW-149426.
00199         valueDivScalar = ((T) 1000000.0);
00200         valueStringCopy.Erase(valueString.Length()-1, 1);
00201         break;
00202     }
00203
00204     bool success = AAX_IDisplayDelegateDecorator<T>::StringToValue(valueStringCopy, value);
00205     *value = valueScalar * (*value);
00206     *value = (*value) / valueDivScalar;
00207     return success;
00208 }
00209
00210
00211 #endif //AAX_CUNITPREFIXDISPLAYDELEGATEDECORATOR

```

15.125 AAX_EndianSwap.h File Reference

```
#include <algorithm>
```

15.125.1 Description

Utility functions for byte-swapping. Used by [AAX_CChunkDataParser](#).

Macros

- #define [ENDIANSWAP_H](#)

Functions

- `template<class T >`
`void AAX_EndianSwapInPlace (T *theDataP)`
Byte swap data in-place.
- `template<class T >`
`T AAX_EndianSwap (T theData)`
Make a byte-swapped copy of data.
- `template<class T >`
`void AAX_BigEndianNativeSwapInPlace (T *theDataP)`
Convert data in-place between Big Endian and native byte ordering.
- `template<class T >`
`T AAX_BigEndianNativeSwap (T theData)`
Copy and convert data between Big Endian and native byte ordering.
- `template<class T >`
`void AAX_LittleEndianNativeSwapInPlace (T *theDataP)`
Convert data in-place from the native byte ordering to Little Endian byte ordering.
- `template<class T >`
`T AAX_LittleEndianNativeSwap (T theData)`
Copy and convert data from the native byte ordering to Little Endian byte ordering.
- `template<class Iter >`
`void AAX_EndianSwapSequenceInPlace (Iter beginI, Iter endI)`
Byte swap a sequence of data in-place.
- `template<class Iter >`
`void AAX_BigEndianNativeSwapSequenceInPlace (Iter beginI, Iter endI)`
Convert an sequence of data in-place between Big Endian and native byte ordering.
- `template<class Iter >`
`void AAX_LittleEndianNativeSwapSequenceInPlace (Iter beginI, Iter endI)`
Convert an sequence of data in-place from the native byte ordering to Little Endian byte ordering.

15.125.2 Macro Definition Documentation

15.125.2.1 ENDIANSWAP_H

```
#define ENDIANSWAP_H
```

15.125.3 Function Documentation

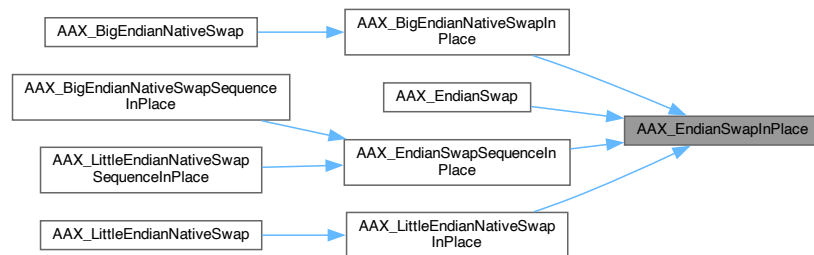
15.125.3.1 AAX_EndianSwapInPlace()

```
template<class T >
void AAX_EndianSwapInPlace (
    T * theDataP ) [inline]
```

Byte swap data in-place.

Referenced by [AAX_BigEndianNativeSwapInPlace\(\)](#), [AAX_EndianSwap\(\)](#), [AAX_EndianSwapSequenceInPlace\(\)](#), and [AAX_LittleEndianNativeSwapInPlace\(\)](#).

Here is the caller graph for this function:



15.125.3.2 AAX_EndianSwap()

```
template<class T >
T AAX_EndianSwap (
    T theData ) [inline]
```

Make a byte-swapped copy of data.

References [AAX_EndianSwapInPlace\(\)](#).

Here is the call graph for this function:



15.125.3.3 AAX_BigEndianNativeSwapInPlace()

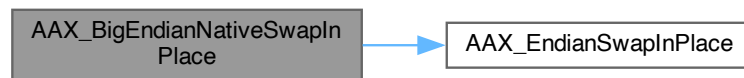
```
template<class T >
void AAX_BigEndianNativeSwapInPlace (
    T * theDataP ) [inline]
```

Convert data in-place between Big Endian and native byte ordering.

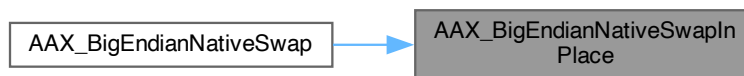
References [AAX_EndianSwapInPlace\(\)](#).

Referenced by [AAX_BigEndianNativeSwap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.125.3.4 AAX_BigEndianNativeSwap()

```
template<class T >
T AAX_BigEndianNativeSwap (
    T theData ) [inline]
```

Copy and convert data between Big Endian and native byte ordering.

References [AAX_BigEndianNativeSwapInPlace\(\)](#).

Here is the call graph for this function:



15.125.3.5 AAX_LittleEndianNativeSwapInPlace()

```
template<class T >
void AAX_LittleEndianNativeSwapInPlace (
    T * theDataP ) [inline]
```

Convert data in-place from the native byte ordering to Little Endian byte ordering.

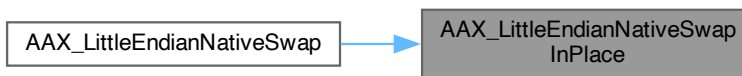
References [AAX_EndianSwapInPlace\(\)](#).

Referenced by [AAX_LittleEndianNativeSwap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.125.3.6 AAX_LittleEndianNativeSwap()

```
template<class T >
T AAX_LittleEndianNativeSwap (
    T theData ) [inline]
```

Copy and convert data from the native byte ordering to Little Endian byte ordering.

References [AAX_LittleEndianNativeSwapInPlace\(\)](#).

Here is the call graph for this function:



15.125.3.7 AAX_EndianSwapSequenceInPlace()

```
template<class Iter >
void AAX_EndianSwapSequenceInPlace (
    Iter beginI,
    Iter endI ) [inline]
```

Byte swap a sequence of data in-place.

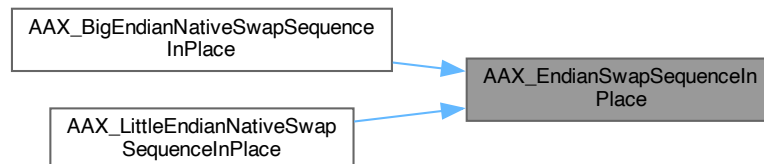
References [AAX_EndianSwapInPlace\(\)](#).

Referenced by [AAX_BigEndianNativeSwapSequenceInPlace\(\)](#), and [AAX_LittleEndianNativeSwapSequenceInPlace\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



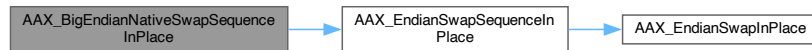
15.125.3.8 AAX_BigEndianNativeSwapSequenceInPlace()

```
template<class Iter >
void AAX_BigEndianNativeSwapSequenceInPlace (
    Iter beginI,
    Iter endI ) [inline]
```

Convert an sequence of data in-place between Big Endian and native byte ordering.

References [AAX_EndianSwapSequenceInPlace\(\)](#).

Here is the call graph for this function:



15.125.3.9 AAX_LittleEndianNativeSwapSequenceInPlace()

```

template<class Iter >
void AAX_LittleEndianNativeSwapSequenceInPlace (
    Iter beginI,
    Iter endI ) [inline]
  
```

Convert an sequence of data in-place from the native byte ordering to Little Endian byte ordering.

References [AAX_EndianSwapSequenceInPlace\(\)](#).

Here is the call graph for this function:



15.126 AAX_EndianSwap.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013–2015, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #pragma once
00022
00023 #ifndef ENDIANSWAP_H
00024 #define ENDIANSWAP_H
00025
00026 // Standard headers
00027 #include <algorithm>
00028
00031 template<class T> void AAX_EndianSwapInPlace(T* theDataP);
00032
00034 template<class T> T AAX_EndianSwap(T theData);
00035
  
```

```

00036
00038 template<class T> void AAX_BigEndianNativeSwapInPlace(T* theDataP);
00039
00041 template<class T> T AAX_BigEndianNativeSwap(T theData);
00042
00044 template<class T> void AAX_LittleEndianNativeSwapInPlace(T* theDataP);
00045
00047 template<class T> T AAX_LittleEndianNativeSwap(T theData);
00048
00049
00050
00052 template<class Iter> void AAX_EndianSwapSequenceInPlace(Iter beginI, Iter endI);
00053
00054
00056 template<class Iter> void AAX_BigEndianNativeSwapSequenceInPlace(Iter beginI, Iter endI);
00057
00059 template<class Iter> void AAX_LittleEndianNativeSwapSequenceInPlace(Iter beginI, Iter endI);
00060
00061
00062 //
00063 // Implementations
00064 //
00065
00066 template<class T>
00067 inline
00068 void
00069 AAX_EndianSwapInPlace(T* theDataP)
00070 {
00071     char *begin, *end;
00072
00073     begin = reinterpret_cast<char*>(theDataP);
00074     end = begin + sizeof( T );
00075     std::reverse( begin, end );
00076 }
00077
00078
00079 template<class T>
00080 inline
00081 T AAX_EndianSwap(T theData)
00082 {
00083     AAX_EndianSwapInPlace(&theData);
00084     return theData;
00085 }
00086
00087
00088 template<class T>
00089 inline
00090 void AAX_BigEndianNativeSwapInPlace(T* theDataP)
00091 {
00092     #if (!defined __BIG_ENDIAN__) || (0 == __BIG_ENDIAN__)
00093         AAX_EndianSwapInPlace(theDataP);
00094     #endif
00095 }
00096
00097
00098 template<class T>
00099 inline
00100 T AAX_BigEndianNativeSwap(T theData)
00101 {
00102     AAX_BigEndianNativeSwapInPlace(&theData);
00103     return theData;
00104 }
00105
00106
00107 template<class T>
00108 inline
00109 void AAX_LittleEndianNativeSwapInPlace(T* theDataP)
00110 {
00111     #if (defined __BIG_ENDIAN__) && (0 != __BIG_ENDIAN__)
00112         AAX_EndianSwapInPlace(theDataP);
00113     #endif
00114 }
00115
00116
00117 template<class T>
00118 inline
00119 T AAX_LittleEndianNativeSwap(T theData)
00120 {
00121     AAX_LittleEndianNativeSwapInPlace(&theData);
00122     return theData;
00123 }
00124
00125
00126 template<class Iter>
00127 inline
00128 void AAX_EndianSwapSequenceInPlace(Iter beginI, Iter endI)
00129 {

```

```

00130     for(Iter i = beginI; i != endI; ++i)
00131     {
00132         // WARNING : Will this give a compile error if a use mistakenly uses it on a const sequence?
00133         AAX_EndianSwapInPlace(&(*i));
00134     };
00135 }
00136
00137
00138 template<class Iter>
00139 inline
00140 void AAX_BigEndianNativeSwapSequenceInPlace(Iter beginI, Iter endI)
00141 {
00142     #if (!defined __BIG_ENDIAN__) || (0 == __BIG_ENDIAN__)
00143         AAX_EndianSwapSequenceInPlace(beginI, endI);
00144     #endif
00145 }
00146
00147
00148 template<class Iter>
00149 inline
00150 void AAX_LittleEndianNativeSwapSequenceInPlace(Iter beginI, Iter endI)
00151 {
00152     #if (defined __BIG_ENDIAN__) && (0 != __BIG_ENDIAN__)
00153         AAX_EndianSwapSequenceInPlace(beginI, endI);
00154     #endif
00155 }
00156
00157 #endif

```

15.127 AAX_Enums.h File Reference

```
#include <stdint.h>
```

15.127.1 Description

Utility functions for byte-swapping. Used by [AAX_CChunkDataParser](#).

Macros

- #define [AAX_INT32_MIN](#) (-2147483647 - 1) /** minimum signed 32 bit value */
 - #define [AAX_INT32_MAX](#) 2147483647 /** maximum signed 32 bit value */
 - #define [AAX_UINT32_MIN](#) 0U /** minimum unsigned 32 bit value */
 - #define [AAX_UINT32_MAX](#) 4294967295U /** maximum unsigned 32 bit value */
 - #define [AAX_INT16_MIN](#) (-32767 - 1) /** minimum signed 16 bit value */
 - #define [AAX_INT16_MAX](#) 32767 /** maximum signed 16 bit value */
 - #define [AAX_UINT16_MIN](#) 0U /** minimum unsigned 16 bit value */
 - #define [AAX_UINT16_MAX](#) 65535U /** maximum unsigned 16 bit value */
 - #define [AAX_ENUM_SIZE_CHECK](#)(x) extern int __enumSizeCheck[2*(sizeof(uint32_t)==sizeof(x)) - 1]
- Macro to ensure enum type consistency across binaries.*
- #define [AAX_STEM_FORMAT](#)(aIndex, aChannelCount) (static_cast<uint32_t>((static_cast<uint16_t>(aIndex) << 16) | ((aChannelCount >= [AAX_UINT16_MIN](#)) && (aChannelCount <= 0xFFFF) ? aChannelCount & 0xFFFF : 0x0000)))
 - #define [AAX_STEM_FORMAT_CHANNEL_COUNT](#)(aStemFormat) (static_cast<uint16_t>(aStemFormat & 0xFFFF))
 - #define [AAX_STEM_FORMAT_INDEX](#)(aStemFormat) (static_cast<int16_t>((aStemFormat >> 16) & 0xFFFF))

Typedefs

- typedef enum [AAX_EParameterType](#) [AAX_EParameterType](#)
FIC stuff that I can't include without DAE library dependence.
- typedef int32_t [AAX_EParameterOrientation](#)
Typedef for a bitfield of [AAX_EParameterOrientationBits](#) values.

Enumerations

- enum [AAX_EHighlightColor](#) {
[AAX_eHighlightColor_Red](#) = 0 ,
[AAX_eHighlightColor_Blue](#) = 1 ,
[AAX_eHighlightColor_Green](#) = 2 ,
[AAX_eHighlightColor_Yellow](#) = 3 ,
[AAX_eHighlightColor_Num](#) }
Highlight color selector.
- enum [AAX_ETracePriorityHost](#) {
[AAX_eTracePriorityHost_None](#) = 0 ,
[AAX_eTracePriorityHost_Critical](#) = 0x10000000 ,
[AAX_eTracePriorityHost_High](#) = 0x08000000 ,
[AAX_eTracePriorityHost_Normal](#) = 0x04000000 ,
[AAX_eTracePriorityHost_Low](#) = 0x02000000 ,
[AAX_eTracePriorityHost_Lowest](#) = 0x01000000 }
Platform-specific tracing priorities.
- enum [AAX_ETracePriorityDSP](#) {
[AAX_eTracePriorityDSP_None](#) = 0 ,
[AAX_eTracePriorityDSP_Assert](#) = 1 ,
[AAX_eTracePriorityDSP_High](#) = 2 ,
[AAX_eTracePriorityDSP_Normal](#) = 3 ,
[AAX_eTracePriorityDSP_Low](#) = 4 }
Platform-specific tracing priorities.
- enum [AAX_EModifiers](#) {
[AAX_eModifiers_None](#) = 0 ,
[AAX_eModifiers_Shift](#) = (1 << 0) ,
[AAX_eModifiers_Control](#) = (1 << 1) ,
[AAX_eModifiers_Option](#) = (1 << 2) ,
[AAX_eModifiers_Command](#) = (1 << 3) ,
[AAX_eModifiers_SecondaryButton](#) = (1 << 4) ,
[AAX_eModifiers_Alt](#) = [AAX_eModifiers_Option](#) ,
[AAX_eModifiers_Cntl](#) = [AAX_eModifiers_Command](#) ,
[AAX_eModifiers_WINKEY](#) = [AAX_eModifiers_Control](#) }
Modifier key definitions used by AAX API.
- enum [AAX_EAudioBufferLength](#) {
[AAX_eAudioBufferLength_Undefined](#) = -1 ,
[AAX_eAudioBufferLength_1](#) = 0 ,
[AAX_eAudioBufferLength_2](#) = 1 ,
[AAX_eAudioBufferLength_4](#) = 2 ,
[AAX_eAudioBufferLength_8](#) = 3 ,
[AAX_eAudioBufferLength_16](#) = 4 ,
[AAX_eAudioBufferLength_32](#) = 5 ,
[AAX_eAudioBufferLength_64](#) = 6 ,
[AAX_eAudioBufferLength_128](#) = 7 ,
[AAX_eAudioBufferLength_256](#) = 8 ,
[AAX_eAudioBufferLength_512](#) = 9 ,
[AAX_eAudioBufferLength_1024](#) = 10 ,
[AAX_eAudioBufferLength_Max](#) = [AAX_eAudioBufferLength_1024](#) }

Generic buffer length definitions.

- enum `AAX_EAudioBufferLengthDSP` {
`AAX_eAudioBufferLengthDSP_Default` = `AAX_eAudioBufferLength_4` ,
`AAX_eAudioBufferLengthDSP_4` = `AAX_eAudioBufferLength_4` ,
`AAX_eAudioBufferLengthDSP_16` = `AAX_eAudioBufferLength_16` ,
`AAX_eAudioBufferLengthDSP_32` = `AAX_eAudioBufferLength_32` ,
`AAX_eAudioBufferLengthDSP_64` = `AAX_eAudioBufferLength_64` ,
`AAX_eAudioBufferLengthDSP_Max` = `AAX_eAudioBufferLengthDSP_64` }

Currently supported processing buffer length definitions for AAX DSP hosts.

- enum `AAE_EAudioBufferLengthNative` {
`AAX_eAudioBufferLengthNative_Min` = `AAX_eAudioBufferLength_32` ,
`AAX_eAudioBufferLengthNative_Max` = `AAX_eAudioBufferLength_Max` }

Processing buffer length definitions for Native AAX hosts.

- enum `AAX_EMaxAudioSuiteTracks` { `AAX_eMaxAudioSuiteTracks` = 48 }

The maximum number of tracks that an AAX host will process in a non-real-time context.

- enum `AAX_EStemFormat` {
`AAX_eStemFormat_Mono` = `AAX_STEM_FORMAT` (0, 1) ,
`AAX_eStemFormat_Stereo` = `AAX_STEM_FORMAT` (1, 2) ,
`AAX_eStemFormat_LCR` = `AAX_STEM_FORMAT` (2, 3) ,
`AAX_eStemFormat_LCRS` = `AAX_STEM_FORMAT` (3, 4) ,
`AAX_eStemFormat_Quad` = `AAX_STEM_FORMAT` (4, 4) ,
`AAX_eStemFormat_5_0` = `AAX_STEM_FORMAT` (5, 5) ,
`AAX_eStemFormat_5_1` = `AAX_STEM_FORMAT` (6, 6) ,
`AAX_eStemFormat_6_0` = `AAX_STEM_FORMAT` (7, 6) ,
`AAX_eStemFormat_6_1` = `AAX_STEM_FORMAT` (8, 7) ,
`AAX_eStemFormat_7_0_SDDS` = `AAX_STEM_FORMAT` (9, 7) ,
`AAX_eStemFormat_7_1_SDDS` = `AAX_STEM_FORMAT` (10, 8) ,
`AAX_eStemFormat_7_0_DTS` = `AAX_STEM_FORMAT` (11, 7) ,
`AAX_eStemFormat_7_1_DTS` = `AAX_STEM_FORMAT` (12, 8) ,
`AAX_eStemFormat_7_0_2` = `AAX_STEM_FORMAT` (20, 9) ,
`AAX_eStemFormat_7_1_2` = `AAX_STEM_FORMAT` (13, 10) ,
`AAX_eStemFormat_5_0_2` = `AAX_STEM_FORMAT` (21, 7) ,
`AAX_eStemFormat_5_1_2` = `AAX_STEM_FORMAT` (22, 8) ,
`AAX_eStemFormat_5_0_4` = `AAX_STEM_FORMAT` (23, 9) ,
`AAX_eStemFormat_5_1_4` = `AAX_STEM_FORMAT` (24, 10) ,
`AAX_eStemFormat_7_0_4` = `AAX_STEM_FORMAT` (25, 11) ,
`AAX_eStemFormat_7_1_4` = `AAX_STEM_FORMAT` (26, 12) ,
`AAX_eStemFormat_7_0_6` = `AAX_STEM_FORMAT` (35, 13) ,
`AAX_eStemFormat_7_1_6` = `AAX_STEM_FORMAT` (36, 14) ,
`AAX_eStemFormat_9_0_4` = `AAX_STEM_FORMAT` (27, 13) ,
`AAX_eStemFormat_9_1_4` = `AAX_STEM_FORMAT` (28, 14) ,
`AAX_eStemFormat_9_0_6` = `AAX_STEM_FORMAT` (29, 15) ,
`AAX_eStemFormat_9_1_6` = `AAX_STEM_FORMAT` (30, 16) ,
`AAX_eStemFormat_Ambi_1_ACN` = `AAX_STEM_FORMAT` (14, 4) ,
`AAX_eStemFormat_Ambi_2_ACN` = `AAX_STEM_FORMAT` (18, 9) ,
`AAX_eStemFormat_Ambi_3_ACN` = `AAX_STEM_FORMAT` (19, 16) ,
`AAX_eStemFormat_Ambi_4_ACN` = `AAX_STEM_FORMAT` (31, 25) ,
`AAX_eStemFormat_Ambi_5_ACN` = `AAX_STEM_FORMAT` (32, 36) ,
`AAX_eStemFormat_Ambi_6_ACN` = `AAX_STEM_FORMAT` (33, 49) ,
`AAX_eStemFormat_Ambi_7_ACN` = `AAX_STEM_FORMAT` (34, 64) ,
`AAX_eStemFormatNum` = 37 ,
`AAX_eStemFormat_None` = `AAX_STEM_FORMAT` (-100, 0) ,
`AAX_eStemFormat_Any` = `AAX_STEM_FORMAT` (-1, 0) ,
`AAX_eStemFormat_INT32_MAX` = `AAX_INT32_MAX` }

Stem format definitions.

- enum `AAX_EPlugInCategory` {
`AAX_ePlugInCategory_None` = 0x00000000 ,

```

AAX_ePluginCategory_EQ = 0x00000001 ,
AAX_ePluginCategory_Dynamics = 0x00000002 ,
AAX_ePluginCategory_PitchShift = 0x00000004 ,
AAX_ePluginCategory_Reverb = 0x00000008 ,
AAX_ePluginCategory_Delay = 0x00000010 ,
AAX_ePluginCategory_Modulation = 0x00000020 ,
AAX_ePluginCategory_Harmonic = 0x00000040 ,
AAX_ePluginCategory_NoiseReduction = 0x00000080 ,
AAX_ePluginCategory_Dither = 0x00000100 ,
AAX_ePluginCategory_SoundField = 0x00000200 ,
AAX_ePluginCategory_HWGenerators = 0x00000400 ,
AAX_ePluginCategory_SWGenerators = 0x00000800 ,
AAX_ePluginCategory_WrappedPlugin = 0x00001000 ,
AAX_EPluginCategory_Effect = 0x00002000 ,
AAX_ePluginCategory_Example = AAX_EPluginCategory_Effect ,
AAX_EPluginCategory_MIDIEffect = 0x00010000 ,
AAX_ePluginCategory_INT32_MAX = AAX_INT32_MAX }

```

Effect category definitions.

- enum `AAX_EPluginStrings` {


```

AAX_ePluginStrings_Analysis = 0 ,
AAX_ePluginStrings_MonoMode = 1 ,
AAX_ePluginStrings_MultiInputMode = 2 ,
AAX_ePluginStrings_RegionByRegionAnalysis = 3 ,
AAX_ePluginStrings_AllSelectedRegionsAnalysis = 4 ,
AAX_ePluginStrings_RegionName = 5 ,
AAX_ePluginStrings_ClipName = 5 ,
AAX_ePluginStrings_Progress = 6 ,
AAX_ePluginStrings_PluginFileName = 7 ,
AAX_ePluginStrings_Preview = 8 ,
AAX_ePluginStrings_Process = 9 ,
AAX_ePluginStrings_Bypass = 10 ,
AAX_ePluginStrings_ClipNameSuffix = 11 ,
AAX_ePluginStrings_INT32_MAX = AAX_INT32_MAX }

```

Effect string identifiers.

- enum `AAX_EMeterOrientation` {


```

AAX_eMeterOrientation_Default = 0 ,
AAX_eMeterOrientation_BottomLeft = AAX_eMeterOrientation_Default ,
AAX_eMeterOrientation_TopRight = 1 ,
AAX_eMeterOrientation_Center = 2 ,
AAX_eMeterOrientation_PhaseDot = 3 }

```

Meter orientation.

- enum `AAX_EMeterBallisticType` {


```

AAX_eMeterBallisticType_Host = 0 ,
AAX_eMeterBallisticType_NoDecay = 1 }

```

Meter ballistics type.

- enum `AAX_EMeterType` {


```

AAX_eMeterType_Input = 0 ,
AAX_eMeterType_Output = 1 ,
AAX_eMeterType_CLGain = 2 ,
AAX_eMeterType_EGGain = 3 ,
AAX_eMeterType_Analysis = 4 ,
AAX_eMeterType_Other = 5 ,
AAX_eMeterType_None = 31 }

```

Meter type.

- enum `AAX_ECurveType` {


```

AAX_eCurveType_None = 0 ,
AAX_eCurveType_EQ = 'AXeq' ,

```

```
AAX_eCurveType_Dynamics = 'AXdy' ,
AAX_eCurveType_Reduction = 'AXdr' }
```

Different Curve Types that can be queried from the Host.

- enum `AAX_EResourceType` {
`AAX_eResourceType_None` = 0 ,
`AAX_eResourceType_PageTable` ,
`AAX_eResourceType_PageTableDir` }

Types of resources that can be added to an Effect's description.

- enum `AAX_ENotificationEvent` {
`AAX_eNotificationEvent_InsertPositionChanged` = 'AXip' ,
`AAX_eNotificationEvent_TrackNameChanged` = 'AXtn' ,
`AAX_eNotificationEvent_TrackUIDChanged` = 'AXtu' ,
`AAX_eNotificationEvent_TrackPositionChanged` = 'AXtp' ,
`AAX_eNotificationEvent_AlgorithmMoved` = 'AXam' ,
`AAX_eNotificationEvent_GUIOpened` = 'AXgo' ,
`AAX_eNotificationEvent_GUIClosed` = 'AXgc' ,
`AAX_eNotificationEvent_ASProcessingState` = 'AXPr' ,
`AAX_eNotificationEvent_ASPreviewState` = 'ASpv' ,
`AAX_eNotificationEvent_SessionBeingOpened` = 'AXso' ,
`AAX_eNotificationEvent_PresetOpened` = 'AXpo' ,
`AAX_eNotificationEvent_EnteringOfflineMode` = 'AXof' ,
`AAX_eNotificationEvent_ExitingOfflineMode` = 'AXox' ,
`AAX_eNotificationEvent_SessionPathChanged` = 'AXsp' ,
`AAX_eNotificationEvent_SignalLatencyChanged` = 'AXsl' ,
`AAX_eNotificationEvent_DelayCompensationState` = 'AXdc' ,
`AAX_eNotificationEvent_CycleCountChanged` = 'AXcc' ,
`AAX_eNotificationEvent_MaxViewSizeChanged` = 'AXws' ,
`AAX_eNotificationEvent_SideChainBeingConnected` = 'AXsc' ,
`AAX_eNotificationEvent_SideChainBeingDisconnected` = 'AXsd' ,
`AAX_eNotificationEvent_NoiseFloorChanged` = 'AXnf' ,
`AAX_eNotificationEvent_ParameterMappingChanged` = 'AXpm' ,
`AAX_eNotificationEvent_ParameterNameChanged` = 'AXpn' ,
`AAX_eNotificationEvent_HostModeChanged` = 'AXHm' ,
`AAX_eNotificationEvent_PriorSettingsInvalid` = 'AXps' ,
`AAX_eNotificationEvent_LogState` = 'AXls' ,
`AAX_eNotificationEvent_TransportStateChanged` = 'AXts' }

Events IDs for AAX notifications.

- enum `AAX_EHostModeBits` {
`AAX_eHostModeBits_None` = 0 ,
`AAX_eHostModeBits_Live` = (1 << 0) }

Host mode.

- enum `AAX_EHostMode` {
`AAX_eHostMode_Show` = `AAX_eHostModeBits_Live` ,
`AAX_eHostMode_Config` = `AAX_eHostModeBits_None` }

DEPRECATED.

- enum `AAX_EPrivateDataOptions` {
`AAX_ePrivateDataOptions_DefaultOptions` = 0 ,
`AAX_ePrivateDataOptions_KeepOnReset` = (1 << 0) ,
`AAX_ePrivateDataOptions_External` = (1 << 1) ,
`AAX_ePrivateDataOptions_Align8` = (1 << 2) ,
`AAX_ePrivateDataOptions_INT32_MAX` = `AAX_INT32_MAX` }

Options for algorithm private data fields.

- enum `AAX_EConstraintLocationMask` {
`AAX_eConstraintLocationMask_None` = 0 ,
`AAX_eConstraintLocationMask_DataModel` = (1 << 0) ,
`AAX_eConstraintLocationMask_DLLChipAffinity` = (1 << 1) }

Property values to describe location constraints placed on the plug-in's algorithm component (ProcessProc)

- enum `AAX_EConstraintTopology` {
`AAX_eConstraintTopology_None` = 0 ,
`AAX_eConstraintTopology_Monolithic` = 1 }

Property values to describe the topology of the plug-in's modules (e.g. data model, GUI.)

- enum `AAX_EComponentInstanceInitAction` {
`AAX_eComponentInstanceInitAction_AddingNewInstance` = 0 ,
`AAX_eComponentInstanceInitAction_RemovingInstance` = 1 ,
`AAX_eComponentInstanceInitAction_ResetInstance` = 2 }

Selector indicating the action that occurred to prompt a component initialization callback.

- enum `AAX_ESampleRateMask` {
`AAX_eSampleRateMask_No` = 0 ,
`AAX_eSampleRateMask_44100` = (1 << 0) ,
`AAX_eSampleRateMask_48000` = (1 << 1) ,
`AAX_eSampleRateMask_88200` = (1 << 2) ,
`AAX_eSampleRateMask_96000` = (1 << 3) ,
`AAX_eSampleRateMask_176400` = (1 << 4) ,
`AAX_eSampleRateMask_192000` = (1 << 5) ,
`AAX_eSampleRateMask_All` = `AAX_INT32_MAX` }

Property values to describe various sample rates.

- enum `AAX_EParameterType` {
`AAX_eParameterType_Discrete` ,
`AAX_eParameterType_Continuous` }

FIC stuff that I can't include without DAE library dependence.

- enum `AAX_EParameterOrientationBits` {
`AAX_eParameterOrientation_Default` = 0 ,
`AAX_eParameterOrientation_BottomMinTopMax` = 0 ,
`AAX_eParameterOrientation_TopMinBottomMax` = 1 ,
`AAX_eParameterOrientation_LeftMinRightMax` = 0 ,
`AAX_eParameterOrientation_RightMinLeftMax` = 2 ,
`AAX_eParameterOrientation_RotarySingleDotMode` = 0 ,
`AAX_eParameterOrientation_RotaryBoostCutMode` = 4 ,
`AAX_eParameterOrientation_RotaryWrapMode` = 8 ,
`AAX_eParameterOrientation_RotarySpreadMode` = 12 ,
`AAX_eParameterOrientation_RotaryLeftMinRightMax` = 0 ,
`AAX_eParameterOrientation_RotaryRightMinLeftMax` = 16 }

Visual Orientation of a parameter.

- enum `AAX_EParameterValueInfoSelector` {
`AAX_ePageTable_EQ_Band_Type` = 0 ,
`AAX_ePageTable_EQ_InCircuitPolarity` = 1 ,
`AAX_ePageTable_UseAlternateControl` = 2 }

Query type selectors for use with `AAX_IEffectParameters::GetParameterValueInfo()`

- enum `AAX_EEQBandTypes` {
`AAX_eEQBandType_HighPass` = 0 ,
`AAX_eEQBandType_LowShelf` = 1 ,
`AAX_eEQBandType_Parametric` = 2 ,
`AAX_eEQBandType_HighShelf` = 3 ,
`AAX_eEQBandType_LowPass` = 4 ,
`AAX_eEQBandType_Notch` = 5 }

Definitions of band types for EQ page table.

- enum `AAX_EEQInCircuitPolarity` {
`AAX_eEQInCircuitPolarity_Enabled` = 0 ,
`AAX_eEQInCircuitPolarity_Bypassed` = 1 ,
`AAX_eEQInCircuitPolarity_Disabled` = 2 }

Definitions for band in/out for EQ page table.

- enum `AAX_EUseAlternateControl` {
`AAX_eUseAlternateControl_No` = 0 ,
`AAX_eUseAlternateControl_Yes` = 1 }

Definitions for Use Alternate Control parameter.

- enum `AAX_EMIDINodeType` {
`AAX_eMIDINodeType_LocalInput` = 0 ,
`AAX_eMIDINodeType_LocalOutput` = 1 ,
`AAX_eMIDINodeType_Global` = 2 ,
`AAX_eMIDINodeType_Transport` = 3 }

MIDI node types.

- enum `AAX_EUpdateSource` {
`AAX_eUpdateSource_Unspecified` = 0 ,
`AAX_eUpdateSource_Parameter` = 1 ,
`AAX_eUpdateSource_Chunk` = 2 ,
`AAX_eUpdateSource_Delay` = 3 }

Source for values passed into `UpdateParameterNormalizedValue()`.

- enum `AAX_EDataInPortType` {
`AAX_eDataInPortType_Unbuffered` = 0 ,
`AAX_eDataInPortType_Buffered` = 1 ,
`AAX_eDataInPortType_Incremental` = 2 }

Property value for whether a data in port should be buffered or not.

- enum `AAX_EFrameRate` {
`AAX_eFrameRate_Undeclared` = 0 ,
`AAX_eFrameRate_24Frame` = 1 ,
`AAX_eFrameRate_25Frame` = 2 ,
`AAX_eFrameRate_2997NonDrop` = 3 ,
`AAX_eFrameRate_2997DropFrame` = 4 ,
`AAX_eFrameRate_30NonDrop` = 5 ,
`AAX_eFrameRate_30DropFrame` = 6 ,
`AAX_eFrameRate_23976` = 7 ,
`AAX_eFrameRate_47952` = 8 ,
`AAX_eFrameRate_48Frame` = 9 ,
`AAX_eFrameRate_50Frame` = 10 ,
`AAX_eFrameRate_5994NonDrop` = 11 ,
`AAX_eFrameRate_5994DropFrame` = 12 ,
`AAX_eFrameRate_60NonDrop` = 13 ,
`AAX_eFrameRate_60DropFrame` = 14 ,
`AAX_eFrameRate_100Frame` = 15 ,
`AAX_eFrameRate_11988NonDrop` = 16 ,
`AAX_eFrameRate_11988DropFrame` = 17 ,
`AAX_eFrameRate_120NonDrop` = 18 ,
`AAX_eFrameRate_120DropFrame` = 19 }

FrameRate types.

- enum `AAX_EFeetFramesRate` {
`AAX_eFeetFramesRate_23976` = 0 ,
`AAX_eFeetFramesRate_24` = 1 ,
`AAX_eFeetFramesRate_25` = 2 }

FeetFramesRate types.

- enum `AAX_EMidiGlobalNodeSelectors` {
`AAX_eMIDIClick` = 1 << 0 ,
`AAX_eMIDIMtc` = 1 << 1 ,
`AAX_eMIDIBeatClock` = 1 << 2 }

The Global MIDI Node Selectors.

- enum `AAX_EPreviewState` {
`AAX_ePreviewState_Stop` = 0 ,
`AAX_ePreviewState_Start` = 1 }

Offline preview states for use with [AAX_eNotificationEvent_ASPreviewState](#).

- enum [AAX_EProcessingState](#) {
[AAX_eProcessingState_StopPass](#) = 2 ,
[AAX_eProcessingState_StartPass](#) = 3 ,
[AAX_eProcessingState_EndPassGroup](#) = 4 ,
[AAX_eProcessingState_BeginPassGroup](#) = 5 ,
[AAX_eProcessingState_Stop](#) = [AAX_eProcessingState_StopPass](#) ,
[AAX_eProcessingState_Start](#) = [AAX_eProcessingState_StartPass](#) }

Offline preview states for use with [AAX_eNotificationEvent_ASProcessingState](#).

- enum [AAX_ETargetPlatform](#) {
[kAAX_eTargetPlatform_None](#) = 0 ,
[kAAX_eTargetPlatform_Native](#) = 1 ,
[kAAX_eTargetPlatform_TI](#) = 2 ,
[kAAX_eTargetPlatform_External](#) = 3 ,
[kAAX_eTargetPlatform_Count](#) = 5 }

Describes what platform the component runs on.

- enum [AAX_ESupportLevel](#) {
[AAX_eSupportLevel_Uninitialized](#) = 0 ,
[AAX_eSupportLevel_Unsupported](#) = 1 ,
[AAX_eSupportLevel_Supported](#) = 2 ,
[AAX_eSupportLevel_Disabled](#) = 3 ,
[AAX_eSupportLevel_ByProperty](#) = 4 }
- enum [AAX_EHostLevel](#) {
[AAX_eHostLevel_Unknown](#) = 0 ,
[AAX_eHostLevel_Standard](#) = 1 ,
[AAX_eHostLevel_Entry](#) = 2 ,
[AAX_eHostLevel_Intermediate](#) = 3 }

Host levels.

- enum [AAX_ETextEncoding](#) {
[AAX_eTextEncoding_Undefined](#) = -1 ,
[AAX_eTextEncoding_UTF8](#) = 0 ,
[AAX_eTextEncoding_Num](#) }

Describes possible string encodings.

- enum [AAX_EAssertFlags](#) {
[AAX_eAssertFlags_Default](#) = 0 ,
[AAX_eAssertFlags_Log](#) = 1 << 0 ,
[AAX_eAssertFlags_Dialog](#) = 1 << 1 }

Flags for use with [AAX_IHostServices::HandleAssertFailure\(\)](#)

- enum [AAX_ETransportState](#) {
[AAX_eTransportState_Unknown](#) = 0 ,
[AAX_eTransportState_Stopping](#) = 1 ,
[AAX_eTransportState_Stop](#) = 2 ,
[AAX_eTransportState_Paused](#) = 3 ,
[AAX_eTransportState_Play](#) = 4 ,
[AAX_eTransportState_FastForward](#) = 5 ,
[AAX_eTransportState_Rewind](#) = 6 ,
[AAX_eTransportState_Scrub](#) = 11 ,
[AAX_eTransportState_Shuttle](#) = 12 ,
[AAX_eTransportState_Num](#) }

Used to indicate the current transport state of the host. This is the global transport state; it does not indicate a track-specific state.

- enum [AAX_ERecordMode](#) {
[AAX_eRecordMode_Unknown](#) = 0 ,
[AAX_eRecordMode_None](#) = 1 ,
[AAX_eRecordMode_Normal](#) = 2 ,
[AAX_eRecordMode_Destructive](#) = 3 ,

```

AAX_eRecordMode_QuickPunch = 4 ,
AAX_eRecordMode_TrackPunch = 5 ,
AAX_eRecordMode_Num }

```

Used to indicate the current record mode of the host. This is the global record mode; it does not indicate a track-specific state.

Functions

- [AAX_ENUM_SIZE_CHECK \(AAX_EHighlightColor\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ETracePriorityHost\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ETracePriorityDSP\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EModifiers\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EAudioBufferLength\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EAudioBufferLengthDSP\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAE_EAudioBufferLengthNative\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EMaxAudioSuiteTracks\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EStemFormat\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EPlugInCategory\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EPlugInStrings\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EMeterOrientation\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EMeterBallisticType\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EMeterType\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ECurveType\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EResourceType\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ENotificationEvent\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EHostModeBits\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EHostMode\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EPrivateDataOptions\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EConstraintLocationMask\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EConstraintTopology\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EComponentInstanceInitAction\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ESampleRateMask\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EParameterType\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EParameterOrientationBits\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EParameterValueInfoSelector\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EEQBandTypes\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EEQInCircuitPolarity\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EUseAlternateControl\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EMIDINodeType\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EUpdateSource\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EDataInPortType\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EFrameRate\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EFeetFramesRate\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EMidiGlobalNodeSelectors\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EPreviewState\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EProcessingState\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ETargetPlatform\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ESupportLevel\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EHostLevel\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ETextEncoding\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_EAssertFlags\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ETransportState\)](#)
- [AAX_ENUM_SIZE_CHECK \(AAX_ERecordMode\)](#)

15.127.2 Macro Definition Documentation

15.127.2.1 AAX_INT32_MIN

```
#define AAX_INT32_MIN (-2147483647 - 1) /** minimum signed 32 bit value */
```

15.127.2.2 AAX_INT32_MAX

```
#define AAX_INT32_MAX 2147483647 /** maximum signed 32 bit value */
```

15.127.2.3 AAX_UINT32_MIN

```
#define AAX_UINT32_MIN 0U /** minimum unsigned 32 bit value */
```

15.127.2.4 AAX_UINT32_MAX

```
#define AAX_UINT32_MAX 4294967295U /** maximum unsigned 32 bit value */
```

15.127.2.5 AAX_INT16_MIN

```
#define AAX_INT16_MIN (-32767 - 1) /** minimum signed 16 bit value */
```

15.127.2.6 AAX_INT16_MAX

```
#define AAX_INT16_MAX 32767 /** maximum signed 16 bit value */
```

15.127.2.7 AAX_UINT16_MIN

```
#define AAX_UINT16_MIN 0U /** minimum unsigned 16 bit value */
```


15.127.2.8 AAX_UINT16_MAX

```
#define AAX_UINT16_MAX 65535U /** maximum unsigned 16 bit value */
```

15.127.2.9 AAX_ENUM_SIZE_CHECK

```
#define AAX_ENUM_SIZE_CHECK(
    x ) extern int __enumSizeCheck[ 2*(sizeof(uint32_t)==sizeof(x)) - 1]
```

Macro to ensure enum type consistency across binaries.

15.127.2.10 AAX_STEM_FORMAT

```
#define AAX_STEM_FORMAT(
    aIndex,
    aChannelCount ) ( static_cast<uint32_t>( ( static_cast<uint16_t>(aIndex) << 16
) | ( (aChannelCount >= AAX_UINT16_MIN) && (aChannelCount <= 0xFFFF) ? aChannelCount & 0xFFFF
: 0x0000 ) ) )
```

15.127.2.11 AAX_STEM_FORMAT_CHANNEL_COUNT

```
#define AAX_STEM_FORMAT_CHANNEL_COUNT(
    aStemFormat ) ( static_cast<uint16_t>( aStemFormat & 0xFFFF ) )
```

15.127.2.12 AAX_STEM_FORMAT_INDEX

```
#define AAX_STEM_FORMAT_INDEX(
    aStemFormat ) ( static_cast<int16_t>( ( aStemFormat >> 16 ) & 0xFFFF ) )
```

15.127.3 Typedef Documentation

15.127.3.1 AAX_EParameterType

```
typedef enum AAX_EParameterType AAX_EParameterType
```

FIC stuff that I can't include without DAE library dependence.

Legacy Porting Notes Values must match unnamed type enum in FicTDMControl.h

Todo FLAGGED FOR REMOVAL

15.127.3.2 AAX_EParameterOrientation

```
typedef int32_t AAX_EParameterOrientation
```

Typedef for a bitfield of [AAX_EParameterOrientationBits](#) values.

15.127.4 Enumeration Type Documentation

15.127.4.1 AAX_EHighlightColor

```
enum AAX_EHighlightColor
```

Highlight color selector.

See also

[AAX_IEffectGUI::SetControlHighlightInfo\(\)](#)

Enumerator

AAX_eHighlightColor_Red	
AAX_eHighlightColor_Blue	
AAX_eHighlightColor_Green	
AAX_eHighlightColor_Yellow	
AAX_eHighlightColor_Num	

15.127.4.2 AAX_ETracePriorityHost

```
enum AAX_ETracePriorityHost
```

Platform-specific tracing priorities.

Use the generic `EAXX_Trace_Priority` in plug-ins for cross-platform tracing (see [AAX_Assert.h](#))

Enumerator

AAX_eTracePriorityHost_None	
AAX_eTracePriorityHost_Critical	
AAX_eTracePriorityHost_High	
AAX_eTracePriorityHost_Normal	
AAX_eTracePriorityHost_Low	
AAX_eTracePriorityHost_Lowest	

15.127.4.3 AAX_ETracePriorityDSP

enum [AAX_ETracePriorityDSP](#)

Platform-specific tracing priorities.

Use the generic `EAXX_Trace_Priority` in plug-ins for cross-platform tracing (see [AAX_Assert.h](#))

Enumerator

<code>AAX_eTracePriorityDSP_None</code>	
<code>AAX_eTracePriorityDSP_Assert</code>	
<code>AAX_eTracePriorityDSP_High</code>	
<code>AAX_eTracePriorityDSP_Normal</code>	
<code>AAX_eTracePriorityDSP_Low</code>	

15.127.4.4 AAX_EModifiers

enum [AAX_EModifiers](#)

Modifier key definitions used by AAX API.

Enumerator

<code>AAX_eModifiers_None</code>	
<code>AAX_eModifiers_Shift</code>	Shift.
<code>AAX_eModifiers_Control</code>	Control on Mac, Winkey/Start on PC.
<code>AAX_eModifiers_Option</code>	Option on Mac, Alt on PC.
<code>AAX_eModifiers_Command</code>	Command on Mac, Ctrl on PC.
<code>AAX_eModifiers_SecondaryButton</code>	Secondary mouse button.
<code>AAX_eModifiers_Alt</code>	Option on Mac, Alt on PC.
<code>AAX_eModifiers_Cntl</code>	Command on Mac, Cntl on PC.
<code>AAX_eModifiers_WINKEY</code>	Control on Mac, WINKEY on PC.

15.127.4.5 AAX_EAudioBufferLength

enum [AAX_EAudioBufferLength](#)

Generic buffer length definitions.

These enum values can be used to calculate literal values as powers of two:

```
(1 << AAX\_eAudioBufferLength\_16) == 16;
```

See also

[AAX_EAudioBufferLengthDSP](#)

[AAE_EAudioBufferLengthNative](#)

Enumerator

AAX_eAudioBufferLength_Undefined	
AAX_eAudioBufferLength_1	
AAX_eAudioBufferLength_2	
AAX_eAudioBufferLength_4	
AAX_eAudioBufferLength_8	
AAX_eAudioBufferLength_16	
AAX_eAudioBufferLength_32	
AAX_eAudioBufferLength_64	
AAX_eAudioBufferLength_128	
AAX_eAudioBufferLength_256	
AAX_eAudioBufferLength_512	
AAX_eAudioBufferLength_1024	
AAX_eAudioBufferLength_Max	Maximum buffer length for ProcessProc processing buffers. Audio buffers for other methods, such as the high-latency render callback for AAX Hybrid or the offline render callback for Host Processor effects, may contain more samples than AAX_eAudioBufferLength_Max.

15.127.4.6 AAX_EAudioBufferLengthDSP

enum [AAX_EAudioBufferLengthDSP](#)

Currently supported processing buffer length definitions for AAX DSP hosts.

AAX DSP decks must support at least these buffer lengths. All AAX DSP algorithm ProcessProcs must support exactly one of these buffer lengths.

See also

[AAX_eProperty_DSP_AudioBufferLength](#)

Enumerator

AAX_eAudioBufferLengthDSP_Default	
AAX_eAudioBufferLengthDSP_4	
AAX_eAudioBufferLengthDSP_16	
AAX_eAudioBufferLengthDSP_32	
AAX_eAudioBufferLengthDSP_64	
AAX_eAudioBufferLengthDSP_Max	

15.127.4.7 AAE_EAudioBufferLengthNative

enum [AAE_EAudioBufferLengthNative](#)

Processing buffer length definitions for Native AAX hosts.

All AAX Native plug-ins must support variable buffer lengths. The buffer lengths that a host will use are constrained by the values in this enum. All Native buffer lengths will be powers of two, as per [AAX_EAudioBufferLength](#)

See also

[AAX_eProperty_DSP_AudioBufferLength](#)

Enumerator

AAX_eAudioBufferLengthNative_Min	Minimum Native buffer length.
AAX_eAudioBufferLengthNative_Max	Maximum Native buffer length.

15.127.4.8 AAX_EMaxAudioSuiteTracks

enum [AAX_EMaxAudioSuiteTracks](#)

The maximum number of tracks that an AAX host will process in a non-real-time context.

See also

[AAX_eProperty_NumberOfInputs](#) and [AAX_eProperty_NumberOfOutputs](#)

Enumerator

AAX_eMaxAudioSuiteTracks	
--------------------------	--

15.127.4.9 AAX_EStemFormat

enum [AAX_EStemFormat](#)

Stem format definitions.

A stem format combines a channel count with a semantic meaning for each channel. Usually this is the speaker or speaker position associated with the data in the channel. The meanings of each channel in each stem format (i.e. channel orders) are listed below.

Not all stem formats are supported by all AAX plug-in hosts. An effect may describe support for any stem format combination which it supports and the host will ignore any configurations which it cannot support.

Note

When defining stem format support in [AAX_IHostProcessor](#) effects do not use stem format properties or values. Instead, use [AAX_eProperty_NumberOfInputs](#) and [AAX_eProperty_NumberOfOutputs](#) with integer channel count values.

See also

- [AAX_eProperty_InputStemFormat](#)
- [AAX_eProperty_OutputStemFormat](#)
- [AAX_eProperty_HybridInputStemFormat](#)
- [AAX_eProperty_HybridOutputStemFormat](#)
- [AAX_eProperty_SideChainStemFormat](#)

Enumerator

AAX_eStemFormat_Mono	M.
AAX_eStemFormat_Stereo	L R.
AAX_eStemFormat_LCR	L C R.
AAX_eStemFormat_LCRS	L C R S.
AAX_eStemFormat_Quad	L R Ls Rs.
AAX_eStemFormat_5_0	L C R Ls Rs.
AAX_eStemFormat_5_1	L C R Ls Rs LFE.
AAX_eStemFormat_6_0	L C R Ls Cs Rs.
AAX_eStemFormat_6_1	L C R Ls Cs Rs LFE.
AAX_eStemFormat_7_0_SDDS	L Lc C Rc R Ls Rs.
AAX_eStemFormat_7_1_SDDS	L Lc C Rc R Ls Rs LFE.
AAX_eStemFormat_7_0_DTS	L C R Lss Rss Lsr Rsr.
AAX_eStemFormat_7_1_DTS	L C R Lss Rss Lsr Rsr LFE.
AAX_eStemFormat_7_0_2	L C R Lss Rss Lsr Rsr Lts Rts.
AAX_eStemFormat_7_1_2	L C R Lss Rss Lsr Rsr LFE Lts Rts.
AAX_eStemFormat_5_0_2	L C R Ls Rs Ltm Rtm.
AAX_eStemFormat_5_1_2	L C R Ls Rs LFE Ltm Rtm.
AAX_eStemFormat_5_0_4	L C R Ls Rs Ltf Rtf Ltr Rtr.
AAX_eStemFormat_5_1_4	L C R Ls Rs LFE Ltf Rtf Ltr Rtr.
AAX_eStemFormat_7_0_4	L C R Lss Rss Lsr Rsr Ltf Rtf Ltr Rtr.
AAX_eStemFormat_7_1_4	L C R Lss Rss Lsr Rsr LFE Ltf Rtf Ltr Rtr.
AAX_eStemFormat_7_0_6	L C R Lss Rss Lsr Rsr Ltf Rtf Ltm Rtm Ltr Rtr.
AAX_eStemFormat_7_1_6	L C R Lss Rss Lsr Rsr LFE Ltf Rtf Ltm Rtm Ltr Rtr.
AAX_eStemFormat_9_0_4	L C R Lw Rw Lss Rss Lsr Rsr Ltf Rtf Ltr Rtr.
AAX_eStemFormat_9_1_4	L C R Lw Rw Lss Rss Lsr Rsr LFE Ltf Rtf Ltr Rtr.
AAX_eStemFormat_9_0_6	L C R Lw Rw Lss Rss Lsr Rsr Ltf Rtf Ltm Rtm Ltr Rtr.
AAX_eStemFormat_9_1_6	L C R Lw Rw Lss Rss Lsr Rsr LFE Ltf Rtf Ltm Rtm Ltr Rtr.
AAX_eStemFormat_Ambi_1_ACN	Ambisonics: first-order with ACN channel order and SN3D (AmbiX) normalization.
AAX_eStemFormat_Ambi_2_ACN	Ambisonics: second-order with ACN channel order and SN3D (AmbiX) normalization.
AAX_eStemFormat_Ambi_3_ACN	Ambisonics: third-order with ACN channel order and SN3D (AmbiX) normalization.
AAX_eStemFormat_Ambi_4_ACN	Ambisonics: fourth-order with ACN channel order and SN3D (AmbiX) normalization.
AAX_eStemFormat_Ambi_5_ACN	Ambisonics: fifth-order with ACN channel order and SN3D (AmbiX) normalization.

Enumerator

AAX_eStemFormat_Ambi_6_ACN	Ambisonics: sixth-order with ACN channel order and SN3D (AmbiX) normalization.
AAX_eStemFormat_Ambi_7_ACN	Ambisonics: seventh-order with ACN channel order and SN3D (AmbiX) normalization.
AAX_eStemFormatNum	
AAX_eStemFormat_None	
AAX_eStemFormat_Any	
AAX_eStemFormat_INT32_MAX	

15.127.4.10 AAX_EPlugInCategory

enum [AAX_EPlugInCategory](#)

Effect category definitions.

Used with [AAX_IEffectDescriptor::AddCategory\(\)](#) to categorize an Effect.

These values are bitwise-exclusive and may be used in a bitmask to define multiple categories:

```
myCategory = AAX_ePlugInCategory_EQ | AAX_ePlugInCategory_Dynamics;
```

Note

The host may handle plug-ins with different categories in different manners, e.g. replacing "analyze" with "reverse" for offline processing of delays and reverbs.

Enumerator

AAX_ePlugInCategory_None	
AAX_ePlugInCategory_EQ	Equalization.
AAX_ePlugInCategory_Dynamics	Compressor, expander, limiter, etc.
AAX_ePlugInCategory_PitchShift	Pitch processing.
AAX_ePlugInCategory_Reverb	Reverberation and room/space simulation.
AAX_ePlugInCategory_Delay	Delay and echo.
AAX_ePlugInCategory_Modulation	Phasing, flanging, chorus, etc.
AAX_ePlugInCategory_Harmonic	Distortion, saturation, and harmonic enhancement.
AAX_ePlugInCategory_NoiseReduction	Noise reduction.
AAX_ePlugInCategory_Dither	Dither, noise shaping, etc.
AAX_ePlugInCategory_SoundField	Pan, auto-pan, upmix and downmix, and surround handling.
AAX_ePlugInCategory_HWGenerators	Fixed hardware audio sources such as SampleCell.
AAX_ePlugInCategory_SWGenerators	Virtual instruments, metronomes, and other software audio sources.
AAX_ePlugInCategory_WrappedPlugin	All plug-ins wrapped by a third party wrapper (i.e. VST to RTAS wrapper), except for VI plug-ins which should be mapped to AAX_PlugInCategory_SWGenerators.
AAX_EPlugInCategory_Effect	Special effects.
AAX_ePlugInCategory_Example	
AAX_EPlugInCategory_MIDIEffect	MIDI effects.
AAX_ePlugInCategory_INT32_MAX	

15.127.4.11 AAX_EPlugInStrings

enum [AAX_EPlugInStrings](#)

Effect string identifiers.

The AAX host may associate certain plug-in display strings with these identifiers.

See also

[AAX_IEffectGUI::GetCustomLabel\(\)](#)

Enumerator

AAX_ePlugInStrings_Analysis	<p>"Analyze" button label (AudioSuite)</p> <p>Legacy Porting Notes Was pluginStrings_Analysis in the RTAS/TDM SDK</p>
AAX_ePlugInStrings_MonoMode	<p>"Mono Mode" selector label (AudioSuite)</p> <p>Legacy Porting Notes Was pluginStrings_MonoMode in the RTAS/TDM SDK</p>
AAX_ePlugInStrings_MultiInputMode	<p>"Multi-Input Mode" selector label (AudioSuite)</p> <p>Legacy Porting Notes Was pluginStrings_Multi↔InputMode in the RTAS/TDM SDK</p>
AAX_ePlugInStrings_RegionByRegionAnalysis	<p>"Clip-by-Clip Analysis" selector label (AudioSuite)</p> <p>Legacy Porting Notes Was pluginStrings_Region↔ByRegionAnalysis in the RTAS/TDM SDK</p>
AAX_ePlugInStrings_AllSelectedRegionsAnalysis	<p>"Whole File Analysis" selector label (AudioSuite)</p> <p>Legacy Porting Notes Was pluginStrings_All↔SelectedRegionsAnalysis in the RTAS/TDM SDK</p>
AAX_ePlugInStrings_RegionName	<p>Deprecated</p>
AAX_ePlugInStrings_ClipName	<p>Clip name label (AudioSuite). This value will replace the clip's name.</p> <p>See also</p> <p>AAX_ePlugInStrings_ClipNameSuffix</p> <p>Legacy Porting Notes Was pluginStrings_RegionName in the RTAS/TDM SDK</p>

Enumerator

AAX_ePlugInStrings_Progress	Progress bar label (AudioSuite) Host Compatibility Notes Not currently supported by Pro Tools Legacy Porting Notes Was pluginStrings_Progress in the RTAS/TDM SDK
AAX_ePlugInStrings_PluginFileName	Deprecated
AAX_ePlugInStrings_Preview	Deprecated
AAX_ePlugInStrings_Process	"Render" button label (AudioSuite) Legacy Porting Notes Was pluginStrings_Process in the RTAS/TDM SDK
AAX_ePlugInStrings_Bypass	"Bypass" button label (AudioSuite) Legacy Porting Notes Was pluginStrings_Bypass in the RTAS/TDM SDK
AAX_ePlugInStrings_ClipNameSuffix	Clip name label suffix (AudioSuite). This value will be appended to the clip's name, vs AAX_ePlugInStrings_ClipName which will replace the clip's name completely.
AAX_ePlugInStrings_INT32_MAX	

15.127.4.12 AAX_EMeterOrientation

enum [AAX_EMeterOrientation](#)

Meter orientation.

Use this enum in conjunction with the [AAX_eProperty_Meter_Orientation](#) property

For more information about meters in [AAX](#), see [Plug-in meters](#)

Enumerator

AAX_eMeterOrientation_Default	
AAX_eMeterOrientation_BottomLeft	the default orientation
AAX_eMeterOrientation_TopRight	Some dynamics plug-in orient their gain reduction like so.
AAX_eMeterOrientation_Center	A plug-in that does gain increase and decrease may want this. meter values less than 0x40000000 would display downward from the mid-point. meter values greater than 0x40000000 would display upward from the mid-point.
AAX_eMeterOrientation_PhaseDot	linear scale, displays 2 dots around the value (currently D-Control only)

15.127.4.13 AAX_EMeterBallisticType

enum [AAX_EMeterBallisticType](#)

Meter ballistics type.

Use this enum in conjunction with the [AAX_eProperty_Meter_Ballistics](#) property

For more information about meters in [AAX](#), see [Plug-in meters](#)

Enumerator

AAX_eMeterBallisticType_Host	The ballistics follow the host settings.
AAX_eMeterBallisticType_NoDecay	No decay ballistics.

15.127.4.14 AAX_EMeterType

enum [AAX_EMeterType](#)

Meter type.

Use this enum in conjunction with the [AAX_eProperty_Meter_Type](#) property

For more information about meters in [AAX](#), see [Plug-in meters](#)

Enumerator

AAX_eMeterType_Input	e.g. Your typical input meter (possibly after an input gain stage)
AAX_eMeterType_Output	e.g. Your typical output meter (possibly after an output gain stage)
AAX_eMeterType_CLGain	e.g. Compressor/Limiter gain reduction
AAX_eMeterType_EGGain	e.g. Expander/Gate gain reduction
AAX_eMeterType_Analysis	e.g. multi-band amplitude from a Spectrum analyzer
AAX_eMeterType_Other	e.g. a meter that does not fit in any of the above categories
AAX_eMeterType_None	For internal host use only.

15.127.4.15 AAX_EResourceType

enum [AAX_EResourceType](#)

Types of resources that can be added to an Effect's description.

See also

[AAX_IEffectDescriptor::AddResourceInfo\(\)](#)

Enumerator

AAX_eResourceType_None	
AAX_eResourceType_PageTable	The file name of the page table xml file
AAX_eResourceType_PageTableDir	The absolute path to the directory containing the plug-in's page table xml file(s) Defaults to *.aaxplugin/Contents/Resources

15.127.4.16 AAX_ENotificationEvent

enum [AAX_ENotificationEvent](#)

Events IDs for AAX notifications.

- Notifications listed with *Sent by: Host* are dispatched by the AAX host and may be received in one or more of
- [AAX_IEffectParameters::NotificationReceived\(\)](#)
- [AAX_IEffectGUI::NotificationReceived\(\)](#)
- [AAX_IEffectDirectData::NotificationReceived\(\)](#)

The host will choose which components are registered to receive each event type. See the documentation for each event type for more information.

Note

All 'AX__' four-char IDs are reserved for the AAX specification

Enumerator

AAX_eNotificationEvent_InsertPositionChanged	(not currently sent) The zero-indexed insert position of this plug-in instance within its track <i>Data: int32_t</i> <i>Sent by: Host</i>
AAX_eNotificationEvent_TrackNameChanged	(const AAX_IString) The current name of this plug-in instance's track Host Compatibility Notes Supported in Pro Tools 11.2 and higher Not supported by Media Composer <i>Data: const AAX_IString</i> <i>Sent by: Host</i>
AAX_eNotificationEvent_TrackUIDChanged	(not currently sent) The current UID of this plug-in instance's track <i>Data: const uint8_t[16]</i> <i>Sent by: Host</i>
AAX_eNotificationEvent_TrackPositionChanged	(not currently sent) The current position index of this plug-in instance's track <i>Data: int32_t</i> <i>Sent by: Host</i>

Enumerator

AAX_eNotificationEvent_AlgorithmMoved	Not currently sent. <i>Data: none</i> <i>Sent by: Host</i>
AAX_eNotificationEvent_GUIOpened	Not currently sent. <i>Data: none</i> <i>Sent by: Host</i>
AAX_eNotificationEvent_GUIClosed	Not currently sent. <i>Data: none</i> <i>Sent by: Host</i>
AAX_eNotificationEvent_ASProcessingState	<p>AudioSuite processing state change notification. One of AAX_EProcessingState.</p> <p>Host Compatibility Notes Supported in Pro Tools 11 and higher Not supported by Media Composer</p> <p><i>Data: int32_t</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_ASPreviewState	<p>AudioSuite preview state change notification. One of AAX_EPreviewState.</p> <p>Legacy Porting Notes Replacement for <code>SetPreviewState()</code></p> <p>Host Compatibility Notes Supported in Pro Tools 11 and higher Not supported by Media Composer</p> <p><i>Data: int32_t</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_SessionBeingOpened	<p>Tell the plug-in that chunk data is coming from a PTX.</p> <p>Host Compatibility Notes Supported in Pro Tools 11 and higher Not supported by Media Composer</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_PresetOpened	<p>Tell the plug-in that chunk data is coming from a TFX.</p> <p>Host Compatibility Notes Supported in Pro Tools 11 and higher</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_EnteringOfflineMode	<p>Entering offline processing mode (i.e. offline bounce)</p> <p>Host Compatibility Notes Supported in Pro Tools 11 and higher</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>

Enumerator

AAX_eNotificationEvent_ExitingOfflineMode	<p>Exiting offline processing mode (i.e. offline bounce)</p> <p>Host Compatibility Notes Supported in Pro Tools 11 and higher</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_SessionPathChanged	<p>A string representing the path of the current session.</p> <p>Host Compatibility Notes Supported in Pro Tools 11.1 and higher</p> <p><i>Data: const AAX_IString</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_SignalLatencyChanged	<p>The host has changed its latency compensation for this plug-in instance.</p> <p>Note</p> <p>This notification may be sent redundantly just after plug-in instantiation when the AAX_eProperty_LatencyContribution property is described.</p> <p>Host Compatibility Notes Supported in Pro Tools 11.1 and higher</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_DelayCompensationState	<p>The host's delay compensation state has changed. This notification refers to the host's delay compensation feature as a whole, rather than the specific delay compensation state for the plug-in. Possible values: 0 (disabled), 1 (enabled) Plug-ins may need to monitor the host's delay compensation state because, while delay compensation is disabled, the host will never change the plug-in's accounted latency and, therefore, will never dispatch AAX_eNotificationEvent_SignalLatencyChanged to the plug-in following a call to AAX_IController::SetSignalLatency().</p> <p>Host Compatibility Notes Supported in Pro Tools 12.6 and higher</p> <p><i>Data: int32_t</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_CycleCountChanged	<p>(not currently sent) The host has changed its DSP cycle allocation for this plug-in instance <i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_MaxViewSizeChanged	<p>Tell the plug-in the maximum allowed GUI dimensions.</p> <p>Host Compatibility Notes Supported in Pro Tools 11.1 and higher</p> <p><i>Data: const AAX_Point</i> <i>Sent by: Host</i></p>

Enumerator

AAX_eNotificationEvent_SideChainBeingConnected	<p>Tell the plug-in about connection of the sidechain input.</p> <p>Host Compatibility Notes Supported in Pro Tools 11.1 and higher</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_SideChainBeing↔ Disconnected	<p>Tell the plug-in about disconnection of the sidechain input.</p> <p>Host Compatibility Notes Supported in Pro Tools 11.1 and higher</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_NoiseFloorChanged	<p>The plug-in's noise floor level. The notification data is the new absolute noise floor level generated by the plug-in, as amplitude. For example, a plug-in generating a noise floor at -80 dB (amplitude) would provide 0.0001 in the notification data.</p> <p>Signal below the level of the plug-in's noise floor may be ignored by host features such as Dynamic Plug-In Processing, which detect whether or not there is any signal being generated by the plug-in</p> <p><i>Data: double</i> <i>Sent by: Plug-in</i></p>
AAX_eNotificationEvent_ParameterMappingChanged	<p>Notify the host that some aspect of the parameters' mapping has changed. To respond to this notification, the host will call AAX_IEffectParameters::UpdatePageTable() to update its cached page tables.</p> <p><i>Data: none</i> <i>Sent by: Plug-in</i></p>
AAX_eNotificationEvent_ParameterNameChanged	<p>Notify the host that one or more parameters' display names have changed. The payload is the parameter's ID. The payload size must be at least as large as the ID string, including the null termination character, and no larger than the size of the buffer containing the AAX_CParamID .</p> <p>Host Compatibility Notes Supported in Pro Tools 2023.3 and higher</p> <p><i>Data: const AAX_CParamID</i> <i>Sent by: Plug-in</i></p>
AAX_eNotificationEvent_HostModeChanged	<p>Notify the plug-in about Host mode changing.</p> <p>Host Compatibility Notes Supported in Venue 5.6 and higher</p> <p><i>Data: AAX_EHostModeBits</i> <i>Sent by: Host</i></p>

Enumerator

AAX_eNotificationEvent_PriorSettingsInvalid	<p>Previously-saved settings may no longer restore the captured state. Use this notification when a change occurs which may cause a different state to be restored by saved settings, and in particular by a saved setting representing the plug-in's state just prior to the change.</p> <p>For example, a plug-in which restricts certain types of state changes when the host is in AAX_eHostModeBits_Live mode should post an AAX_eNotificationEvent_PriorSettingsInvalid notification when this part of the plug-in state is changed manually by the user; if plug-in settings captured prior to this manual change are later set on the plug-in while the host is in live mode then some part of the settings change will be blocked and the captured state will not be perfectly restored.</p> <p>Host Compatibility Notes Supported in Venue 5.6 and higher</p> <p><i>Data: none</i> <i>Sent by: Plug-in</i></p>
AAX_eNotificationEvent_LogState	<p>Notify plug-in to log current state. Plug-in implementation specific</p> <p>Host Compatibility Notes Pro Tools currently only sends this notification to the Direct Data object in the plug-in</p> <p><i>Data: none</i> <i>Sent by: Host</i></p>
AAX_eNotificationEvent_TransportStateChanged	<p>Notify plug-in that the TransportState was changed.</p> <p>Host Compatibility Notes Supported in Pro Tools 2021.10 and higher</p> <p><i>Data: AAX_TransportStateInfo_V1</i> <i>Sent by: Host</i></p>

15.127.4.17 AAX_EHostModeBits

enum [AAX_EHostModeBits](#)

Host mode.

Host Compatibility Notes Supported in Venue 5.6 and higher

Enumerator

AAX_eHostModeBits_None	No special host mode, e.g. Pro Tools normal operation, Venue Config mode.
AAX_eHostModeBits_Live	The host is in a live playback mode, e.g. Venue Show mode - inserts are live and must not allow state changes which interrupt audio processing.
Generated by Doxygen	

15.127.4.18 AAX_EHostMode

enum [AAX_EHostMode](#)

DEPRECATED.

Use [AAX_EHostModeBits](#)

Warning

The values of these modes have changed as of AAX SDK 2.3.1 from the definitions originally published in AAX SDK 2.3.0

Deprecated This enum is deprecated and will be removed in a future release.

Enumerator

AAX_eHostMode_Show	Deprecated Use AAX_eHostModeBits_Live
AAX_eHostMode_Config	Deprecated Use AAX_eHostModeBits_None

15.127.4.19 AAX_EPrivateDataOptions

enum [AAX_EPrivateDataOptions](#)

Options for algorithm private data fields.

Enumerator

AAX_ePrivateDataOptions_DefaultOptions	
AAX_ePrivateDataOptions_KeepOnReset	Retain data upon plug-in reset. Warning Not currently implemented. If this functionality is desired, the recommended workaround is to cache the desired private data to be set during AAX_IEffectParameters::ResetFieldData() .
AAX_ePrivateDataOptions_External	Place the block in external memory (internal by default)
AAX_ePrivateDataOptions_Align8	Place the block in mem aligned by 64 bits.
AAX_ePrivateDataOptions_INT32_MAX	

15.127.4.20 AAX_EConstraintLocationMask

enum [AAX_EConstraintLocationMask](#)

Property values to describe location constraints placed on the plug-in's algorithm component (`ProcessProc`)

See also

[AAX_eProperty_Constraint_Location](#)

Enumerator

<code>AAX_eConstraintLocationMask_None</code>	No constraint placed on component's location.
<code>AAX_eConstraintLocationMask_DataModel</code>	This <code>ProcessProc</code> must be co-located with the plug-in's data model object.
<code>AAX_eConstraintLocationMask_DLLChipAffinity</code>	<p>This <code>ProcessProc</code> should be instantiated on the same chip as other effects that use the same DLL.</p> <ul style="list-style-type: none">This constraint is only applicable to DSP algorithms <p>This property should only be used when absolutely required, as it will constrain the DSP manager and reduce overall DSP plug-in instance counts on the system.</p> <p>Host Compatibility Notes This constraint is supported in Pro Tools 10.2 and higher</p>

15.127.4.21 AAX_EConstraintTopology

enum [AAX_EConstraintTopology](#)

Property values to describe the topology of the plug-in's modules (e.g. data model, GUI.)

See also

[AAX_eProperty_Constraint_Topology](#)

Enumerator

<code>AAX_eConstraintTopology_None</code>	No constraint placed on plug-in's topology.
<code>AAX_eConstraintTopology_Monolithic</code>	All plug-in modules (e.g. data model, GUI) must be co-located and non-relocatable.

15.127.4.22 AAX_EComponentInstanceInitAction

enum [AAX_EComponentInstanceInitAction](#)

Selector indicating the action that occurred to prompt a component initialization callback.

See also

[AAX_CInstanceInitProc](#)

Enumerator

AAX_eComponentInstanceInitAction_AddingNewInstance	
AAX_eComponentInstanceInitAction_RemovingInstance	
AAX_eComponentInstanceInitAction_ResetInstance	

15.127.4.23 AAX_ESampleRateMask

enum [AAX_ESampleRateMask](#)

Property values to describe various sample rates.

These values may be used as a bitmask, so e.g. a particular Effect may declare compatibility with [AAX_eSampleRateMask_44100](#) | [AAX_eSampleRateMask_48000](#)

See also

[AAX_eProperty_SampleRate](#)

Enumerator

AAX_eSampleRateMask_No	
AAX_eSampleRateMask_44100	
AAX_eSampleRateMask_48000	
AAX_eSampleRateMask_88200	
AAX_eSampleRateMask_96000	
AAX_eSampleRateMask_176400	
AAX_eSampleRateMask_192000	
AAX_eSampleRateMask_All	

15.127.4.24 AAX_EParameterType

enum [AAX_EParameterType](#)

FIC stuff that I can't include without DAE library dependence.

Legacy Porting Notes Values must match unnamed type enum in FicTDMControl.h

Todo FLAGGED FOR REMOVAL

Enumerator

AAX_eParameterType_Discrete	Legacy Porting Notes Matches kDAE_DiscreteValues
AAX_eParameterType_Continuous	Legacy Porting Notes Matches kDAE_ContinuousValues

15.127.4.25 AAX_EParameterOrientationBits

enum [AAX_EParameterOrientationBits](#)

Visual Orientation of a parameter.

Todo FLAGGED FOR REVISION

Enumerator

AAX_eParameterOrientation_Default	
AAX_eParameterOrientation_BottomMinTopMax	
AAX_eParameterOrientation_TopMinBottomMax	
AAX_eParameterOrientation_LeftMinRightMax	
AAX_eParameterOrientation_RightMinLeftMax	
AAX_eParameterOrientation_RotarySingleDotMode	
AAX_eParameterOrientation_RotaryBoostCutMode	
AAX_eParameterOrientation_RotaryWrapMode	
AAX_eParameterOrientation_RotarySpreadMode	
AAX_eParameterOrientation_RotaryLeftMinRightMax	
AAX_eParameterOrientation_RotaryRightMinLeftMax	

15.127.4.26 AAX_EParameterValueInfoSelector

enum [AAX_EParameterValueInfoSelector](#)

Query type selectors for use with [AAX_IEffectParameters::GetParameterValueInfo\(\)](#)

See also

[AAX_EEQBandTypes](#)

[AAX_EEQInCircuitPolarity](#)

[AAX_EUseAlternateControl](#)

Legacy Porting Notes converted from `EControlValueInfo` in the legacy SDK

Enumerator

<code>AAX_ePageTable_EQ_Band_Type</code>	EQ filter band type. Possible response values are listed in AAX_EEQBandTypes Legacy Porting Notes converted from <code>eDigi_PageTable_EQ_Band_Type</code> in the legacy SDK
<code>AAX_ePageTable_EQ_InCircuitPolarity</code>	Description of whether a particular EQ band is active. Possible response values are listed in AAX_EEQInCircuitPolarity Legacy Porting Notes converted from <code>eDigi_PageTable_EQ_InCircuitPolarity</code> in the legacy SDK
<code>AAX_ePageTable_UseAlternateControl</code>	Description of whether an alternate parameter should be used for a given slot. For example, some control surfaces support Q/Slope encoders. Using an alternate control mechanism, plug-ins mapped to these devices can assign a different slope control to the alternate slot and have it coexist with a Q control for each band. This is only applicable when mapping separate parameters to the same encoder; if the Q and Slope controls are implemented as the same parameter object in the plug-in then customization is not needed. Possible response values are listed in AAX_EUseAlternateControl Legacy Porting Notes converted from <code>eDigi_PageTable_UseAlternateControl</code> in the legacy SDK

15.127.4.27 AAX_EEQBandTypes

enum [AAX_EEQBandTypes](#)

Definitions of band types for EQ page table.

For the [AAX_ePageTable_EQ_Band_Type](#) parameter value info selector

Enumerator

<code>AAX_eEQBandType_HighPass</code>	Freq, Slope
<code>AAX_eEQBandType_LowShelf</code>	Freq, Gain, Slope
<code>AAX_eEQBandType_Parametric</code>	Freq, Gain, Q
<code>AAX_eEQBandType_HighShelf</code>	Freq, Gain, Slope
<code>AAX_eEQBandType_LowPass</code>	Freq, Slope
<code>AAX_eEQBandType_Notch</code>	Freq, Q

15.127.4.28 AAX_EEQInCircuitPolarity

enum [AAX_EEQInCircuitPolarity](#)

Definitions for band in/out for EQ page table.

For the AAX_ePageTable_EQ_InCircuitPolarity parameter value selector

Enumerator

AAX_eEQInCircuitPolarity_Enabled	EQ band is in the signal path and enabled
AAX_eEQInCircuitPolarity_Bypassed	EQ band is in the signal path but bypassed/off
AAX_eEQInCircuitPolarity_Disabled	EQ band is completely removed from signal path

15.127.4.29 AAX_EUseAlternateControl

enum [AAX_EUseAlternateControl](#)

Definitions for Use Alternate Control parameter.

For the AAX_ePageTable_UseAlternateControl parameter value info selector

Enumerator

AAX_eUseAlternateControl_No	
AAX_eUseAlternateControl_Yes	

15.127.4.30 AAX_EMIDINodeType

enum [AAX_EMIDINodeType](#)

MIDI node types.

See also

[AAX_IComponentDescriptor::AddMIDINode\(\)](#)

Enumerator

AAX_eMIDINodeType_LocalInput	<p>Local MIDI input. Local MIDI input nodes receive MIDI by accessing AAX_CMidiStream buffers filled with MIDI messages. These buffers of MIDI data are available within the algorithm context with data corresponding to the current audio buffer being computed. The Effect can step through this buffer like a "script" to respond to MIDI events within the audio callback.</p> <p>Legacy Porting Notes Corresponds to RTAS Buffered MIDI input nodes in the legacy SDK</p>
AAX_eMIDINodeType_LocalOutput	<p>Local MIDI output. Local MIDI output nodes send MIDI by filling buffers with MIDI messages. Messages posted to MIDI output nodes will be available in the host as MIDI streams, routable to MIDI track inputs and elsewhere.</p> <p>Data posted to a MIDI output buffer will be timed to correspond with the current audio buffer being processed. MIDI outputs support custom timestamping relative to the first sample of the audio buffer.</p> <p>The delivery of variable length SysEx messages is also supported. There are no buffer size limitations for output of SysEx messages. To post a MIDI output buffer, an Effect must construct a series of AAX_CMidiPacket objects and place them in the output buffer provided in the port's AAX_CMidiStream</p> <p>Legacy Porting Notes Corresponds to RTAS Buffered MIDI output nodes in the legacy SDK</p>
AAX_eMIDINodeType_Global	<p>Global MIDI node. Global MIDI nodes allow an Effect to receive streaming global MIDI data like MIDI Time Code, MIDI Beat Clock, and host-specific message formats such as the Click messages used in Pro Tools.</p> <p>The specific kind of data that will be received by a Global MIDI node is specified using a mask of AAX_EMidiGlobalNodeSelectors values. Global MIDI nodes are like local MIDI nodes, except they do not show up as assignable outputs in the host. Instead the MIDI data is automatically routed to the plug-in, without the user making any connections.</p> <p>The buffer of data provided via a Global MIDI node may be shared between all currently active Effect instances, and this node may include both explicitly requested data and data not requested by the current Effect. For example, if one plug-in requests MTC and another plug-in requests Click, all plug-ins connected to this global node will get both MTC and Click messages in the shared buffer.</p> <p>Legacy Porting Notes Corresponds to RTAS Shared Buffer global nodes in the legacy SDK</p>
AAX_eMIDINodeType_Transport	<p>Transport node. Call AAX_IMIDINode::GetTransport() on this node to access the AAX_ITransport interface.</p> <p>Warning</p> <p>See warning at AAX_IMIDINode::GetTransport() regarding use of this interface</p>

15.127.4.31 AAX_EUpdateSource

enum [AAX_EUpdateSource](#)

Source for values passed into [UpdateParameterNormalizedValue\(\)](#).

Enumerator

AAX_eUpdateSource_Unspecified	Parameter updates of unknown / unspecified origin, currently including all updates from control surfaces, GUI edit events, and edits originating in the plug-in outside of the context of UpdateParameterNormalizedValue() or SetChunk() .
AAX_eUpdateSource_Parameter	Parameter updates originating (via AAX_IAutomationDelegate::PostSetValueRequest()) within the scope of UpdateParameterNormalizedValue() .
AAX_eUpdateSource_Chunk	Parameter updates originating (via AAX_IAutomationDelegate::PostSetValueRequest()) within the scope of SetChunk() .
AAX_eUpdateSource_Delay	:Not Used by AAX Plug-Ins

15.127.4.32 AAX_EDataInPortType

enum [AAX_EDataInPortType](#)

Property value for whether a data in port should be buffered or not.

See also

[AAX_IComponentDescriptor::AddDataInPort\(\)](#)

Enumerator

AAX_eDataInPortType_Unbuffered	Data port is unbuffered; the most recently posted packet is always delivered to the alg proc
AAX_eDataInPortType_Buffered	Data port is buffered both on the host and DSP and packets are updated to the current timestamp with every alg proc call Data delivered to alg proc always reflects the latest posted packet that has a timestamp at or before the current processing buffer
AAX_eDataInPortType_Incremental	Data port is buffered both on the host and DSP and packets are updated only once per alg proc call Since only one packet is delivered at a time, all packets will be delivered to the alg proc unless an internal buffer overflow occurs Note If multiple packets are posted to this port <i>before</i> the initial call to the alg proc, only the latest packet will be delivered to the first call to the alg proc. Thereafter, all packets will be delivered incrementally. Host Compatibility Notes Supported in Pro Tools 12.5 and higher; when AAX_eDataInPortType_Incremental is not supported the port will be treated as AAX_eDataInPortType_Unbuffered
Generated by Doxygen	

15.127.4.33 AAX_EFrameRate

enum [AAX_EFrameRate](#)

FrameRate types.

See also

[AAX_ITransport::GetTimeCodeInfo\(\)](#)

[AAX_ITransport::GetHDTTimeCodeInfo\(\)](#)

Enumerator

AAX_eFrameRate_Undeclared	
AAX_eFrameRate_24Frame	
AAX_eFrameRate_25Frame	
AAX_eFrameRate_2997NonDrop	
AAX_eFrameRate_2997DropFrame	
AAX_eFrameRate_30NonDrop	
AAX_eFrameRate_30DropFrame	
AAX_eFrameRate_23976	
AAX_eFrameRate_47952	
AAX_eFrameRate_48Frame	
AAX_eFrameRate_50Frame	
AAX_eFrameRate_5994NonDrop	
AAX_eFrameRate_5994DropFrame	
AAX_eFrameRate_60NonDrop	
AAX_eFrameRate_60DropFrame	
AAX_eFrameRate_100Frame	
AAX_eFrameRate_11988NonDrop	
AAX_eFrameRate_11988DropFrame	
AAX_eFrameRate_120NonDrop	
AAX_eFrameRate_120DropFrame	

15.127.4.34 AAX_EFeetFramesRate

enum [AAX_EFeetFramesRate](#)

FeetFramesRate types.

See also

[AAX_ITransport::GetFeetFramesInfo\(\)](#)

Enumerator

AAX_eFeetFramesRate_23976	
AAX_eFeetFramesRate_24	
AAX_eFeetFramesRate_25	

15.127.4.35 AAX_EMidiGlobalNodeSelectors

enum [AAX_EMidiGlobalNodeSelectors](#)

The Global MIDI Node Selectors.

These selectors are used in the *channelMask* argument of [AAX_IComponentDescriptor::AddMIDINode\(\)](#) and [AAX_IEffectDescriptor::AddControlMIDINode\(\)](#) to request one or more kinds of global data.

Enumerator

AAX_eMIDIClick	<p>Selector to request click messages. The click messages are special 2-byte messages encoded as follows:</p> <ul style="list-style-type: none"> • Accented click: Note on pitch 0 (0x90 0x00) • Unaccented click: Note on pitch 1 (0x90 0x01) <p>Note</p> <p>No <i>Note Off</i> messages are ever sent. This isn't up-to-spec MIDI data, just a way of encoding click events.</p>
AAX_eMIDIMtc	Selector to request MIDI Time Code (MTC) data. The Standard MIDI Time Code format.
AAX_eMIDIBeatClock	Selector to request MIDI Beat Clock (MBC) messages. This includes Song Position Pointer, Start/Stop/Continue, and Midi Clock (F8).

15.127.4.36 AAX_EPreviewState

enum [AAX_EPreviewState](#)

Offline preview states for use with [AAX_eNotificationEvent_ASPreviewState](#).

Note

Do not perform any non-trivial processing within the notification handler. Instead, use the processing state notification to inform the processing that is performed in methods such as [PreRender\(\)](#).

Enumerator

AAX_ePreviewState_Stop	Offline preview has ended. For Host Processor plug-ins, this notification is sent just before the final call to PostRender() , or after analysis is complete for plug-ins with analysis-only preview.
AAX_ePreviewState_Start	Offline preview is beginning. For Host Processor plug-ins, this notification is sent before any calls to PreAnalyze() or to PreRender() .

15.127.4.37 AAX_EProcessingState

enum [AAX_EProcessingState](#)

Offline preview states for use with [AAX_eNotificationEvent_ASProcessingState](#).

Note

Do not perform any non-trivial processing within the notification handler. Instead, use the processing state notification to inform the processing that is performed in methods such as [PreRender\(\)](#).

Enumerator

AAX_eProcessingState_StopPass	A single offline processing pass has ended. A single offline processing pass is an analysis and/or render applied to a set of channels in parallel. For Host Processor plug-ins, this notification is sent just before the final call to PostRender() , or after analysis is complete for analysis-only offline plug-ins.
AAX_eProcessingState_StartPass	A single offline processing pass is beginning. A single offline processing pass is an analysis and/or render applied to a set of channels in parallel. For Host Processor plug-ins, this notification is sent before any calls to PreAnalyze() , PreRender() , or InitOutputBounds() for each processing pass.
AAX_eProcessingState_EndPassGroup	An offline processing pass group has completed. An offline processing pass group is a full set of analysis and/or render passes applied to the complete set of input channels. Host Compatibility Notes AudioSuite pass group notifications are supported starting in Pro Tools 12.0
AAX_eProcessingState_BeginPassGroup	An offline processing pass group is beginning. An offline processing pass group is a full set of analysis and/or render passes applied to the complete set of input channels. Host Compatibility Notes AudioSuite pass group notifications are supported starting in Pro Tools 12.0
AAX_eProcessingState_Stop	Deprecated
AAX_eProcessingState_Start	
	Deprecated

15.127.4.38 AAX_ETargetPlatform

enum [AAX_ETargetPlatform](#)

Describes what platform the component runs on.

Enumerator

kAAX_eTargetPlatform_None	
kAAX_eTargetPlatform_Native	
kAAX_eTargetPlatform_TI	
kAAX_eTargetPlatform_External	
kAAX_eTargetPlatform_Count	

15.127.4.39 AAX_ESupportLevel

enum [AAX_ESupportLevel](#)

Feature support indicators

See also

[AAX_IDescriptionHost::AcquireFeatureProperties\(\)](#)

Note

: There is no value defined for unknown features. Instead, unknown features are indicated by [AcquireFeatureProperties\(\)](#) providing a null [AAX_IFeatureInfo](#) in response to a request using the unknown feature UID

Enumerator

AAX_eSupportLevel_Uninitialized	An uninitialized AAX_ESupportLevel
AAX_eSupportLevel_Unsupported	The feature is known but explicitly not supported
AAX_eSupportLevel_Supported	
AAX_eSupportLevel_Disabled	
AAX_eSupportLevel_ByProperty	

15.127.4.40 AAX_EHostLevel

enum [AAX_EHostLevel](#)

Host levels.

Some AAX software hosts support different levels which are sold as separate products. For example, there may be an entry-level version of a product as well as a full version.

The level of a host may impact the user experience, workflows, or the availability of certain plug-ins. For example, some entry-level hosts are restricted to loading only specific plug-ins.

Typically an AAX plug-in should not need to query this information or change its behavior based on the level of the host.

See also

[AAXATTR_Client_Level](#)

Enumerator

AAX_eHostLevel_Unknown	
AAX_eHostLevel_Standard	Standard host level.
AAX_eHostLevel_Entry	Entry-level host.
AAX_eHostLevel_Intermediate	Intermediate-level host.

15.127.4.41 AAX_ETextEncoding

enum [AAX_ETextEncoding](#)

Describes possible string encodings.

Enumerator

AAX_eTextEncoding_Undefined	
AAX_eTextEncoding_UTF8	UTF-8 string encoding.
AAX_eTextEncoding_Num	

15.127.4.42 AAX_EAssertFlags

enum [AAX_EAssertFlags](#)

Flags for use with [AAX_IHostServices::HandleAssertFailure\(\)](#)

Enumerator

AAX_eAssertFlags_Default	No special handler requested.
AAX_eAssertFlags_Log	Logging requested.
AAX_eAssertFlags_Dialog	User-visible modal alert dialog requested.

15.127.4.43 AAX_ETransportState

enum [AAX_ETransportState](#)

Used to indicate the current transport state of the host. This is the global transport state; it does not indicate a track-specific state.

Enumerator

AAX_eTransportState_Unknown	
AAX_eTransportState_Stopping	
AAX_eTransportState_Stop	
AAX_eTransportState_Paused	
AAX_eTransportState_Play	
AAX_eTransportState_FastForward	
AAX_eTransportState_Rewind	
AAX_eTransportState_Scrub	
AAX_eTransportState_Shuttle	
AAX_eTransportState_Num	

15.127.4.44 AAX_ERecordMode

enum [AAX_ERecordMode](#)

Used to indicate the current record mode of the host. This is the global record mode; it does not indicate a track-specific state.

Enumerator

AAX_eRecordMode_Unknown	
AAX_eRecordMode_None	
AAX_eRecordMode_Normal	
AAX_eRecordMode_Destructive	
AAX_eRecordMode_QuickPunch	
AAX_eRecordMode_TrackPunch	
AAX_eRecordMode_Num	

15.127.5 Function Documentation

15.127.5.1 AAX_ENUM_SIZE_CHECK() [1/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EHighlightColor )
```

15.127.5.2 AAX_ENUM_SIZE_CHECK() [2/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ETracePriorityHost )
```

15.127.5.3 AAX_ENUM_SIZE_CHECK() [3/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ETracePriorityDSP )
```

15.127.5.4 AAX_ENUM_SIZE_CHECK() [4/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EModifiers )
```

15.127.5.5 AAX_ENUM_SIZE_CHECK() [5/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EAudioBufferLength )
```

15.127.5.6 AAX_ENUM_SIZE_CHECK() [6/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EAudioBufferLengthDSP )
```

15.127.5.7 AAX_ENUM_SIZE_CHECK() [7/45]

```
AAX_ENUM_SIZE_CHECK (
    AAE_EAudioBufferLengthNative )
```

15.127.5.8 AAX_ENUM_SIZE_CHECK() [8/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EMaxAudioSuiteTracks )
```

15.127.5.9 AAX_ENUM_SIZE_CHECK() [9/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EStemFormat )
```

15.127.5.10 AAX_ENUM_SIZE_CHECK() [10/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EPlugInCategory )
```

15.127.5.11 AAX_ENUM_SIZE_CHECK() [11/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EPlugInStrings )
```

15.127.5.12 AAX_ENUM_SIZE_CHECK() [12/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EMeterOrientation )
```

15.127.5.13 AAX_ENUM_SIZE_CHECK() [13/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EMeterBallisticType )
```

15.127.5.14 AAX_ENUM_SIZE_CHECK() [14/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EMeterType )
```

15.127.5.15 AAX_ENUM_SIZE_CHECK() [15/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ECurveType )
```

15.127.5.16 AAX_ENUM_SIZE_CHECK() [16/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EResourceType )
```

15.127.5.17 AAX_ENUM_SIZE_CHECK() [17/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ENotificationEvent )
```

15.127.5.18 AAX_ENUM_SIZE_CHECK() [18/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EHostModeBits )
```

15.127.5.19 AAX_ENUM_SIZE_CHECK() [19/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EHostMode )
```

15.127.5.20 AAX_ENUM_SIZE_CHECK() [20/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EPrivateDataOptions )
```

15.127.5.21 AAX_ENUM_SIZE_CHECK() [21/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EConstraintLocationMask )
```


15.127.5.22 AAX_ENUM_SIZE_CHECK() [22/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EConstraintTopology )
```

15.127.5.23 AAX_ENUM_SIZE_CHECK() [23/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EComponentInstanceInitAction )
```

15.127.5.24 AAX_ENUM_SIZE_CHECK() [24/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ESampleRateMask )
```

15.127.5.25 AAX_ENUM_SIZE_CHECK() [25/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EParameterType )
```

15.127.5.26 AAX_ENUM_SIZE_CHECK() [26/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EParameterOrientationBits )
```

15.127.5.27 AAX_ENUM_SIZE_CHECK() [27/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EParameterValueInfoSelector )
```

15.127.5.28 AAX_ENUM_SIZE_CHECK() [28/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EEQBandTypes )
```

15.127.5.29 AAX_ENUM_SIZE_CHECK() [29/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EEQInCircuitPolarity )
```

15.127.5.30 AAX_ENUM_SIZE_CHECK() [30/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EUseAlternateControl )
```

15.127.5.31 AAX_ENUM_SIZE_CHECK() [31/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EMIDINodeType )
```

15.127.5.32 AAX_ENUM_SIZE_CHECK() [32/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EUpdateSource )
```

15.127.5.33 AAX_ENUM_SIZE_CHECK() [33/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EDataInPortType )
```

15.127.5.34 AAX_ENUM_SIZE_CHECK() [34/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EFrameRate )
```

15.127.5.35 AAX_ENUM_SIZE_CHECK() [35/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EFeetFramesRate )
```

15.127.5.36 AAX_ENUM_SIZE_CHECK() [36/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EmidiGlobalNodeSelectors )
```

15.127.5.37 AAX_ENUM_SIZE_CHECK() [37/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EPreviewState )
```

15.127.5.38 AAX_ENUM_SIZE_CHECK() [38/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EProcessingState )
```

15.127.5.39 AAX_ENUM_SIZE_CHECK() [39/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ETargetPlatform )
```

15.127.5.40 AAX_ENUM_SIZE_CHECK() [40/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ESupportLevel )
```

15.127.5.41 AAX_ENUM_SIZE_CHECK() [41/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EHostLevel )
```

15.127.5.42 AAX_ENUM_SIZE_CHECK() [42/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ETextEncoding )
```

15.127.5.43 AAX_ENUM_SIZE_CHECK() [43/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_EAssertFlags )
```

15.127.5.44 AAX_ENUM_SIZE_CHECK() [44/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ETransportState )
```

15.127.5.45 AAX_ENUM_SIZE_CHECK() [45/45]

```
AAX_ENUM_SIZE_CHECK (
    AAX_ERecordMode )
```

15.128 AAX_Enums.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019-2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011 */
00012
00019 /*=====*/
00020
00021
00023 #ifndef AAX_ENUMS_H
00024 #define AAX_ENUMS_H
00026
00027 #include <stdint.h>
00028
00029 #define AAX_INT32_MIN (-2147483647 - 1)
00030 #define AAX_INT32_MAX 2147483647
00031 #define AAX_UINT32_MIN 0U
00032 #define AAX_UINT32_MAX 4294967295U
00033 #define AAX_INT16_MIN (-32767 - 1)
00034 #define AAX_INT16_MAX 32767
00035 #define AAX_UINT16_MIN 0U
00036 #define AAX_UINT16_MAX 65535U
00040 #ifndef _TMS320C6X
00041 #define AAX_ENUM_SIZE_CHECK(x) extern int __enumSizeCheck[ 2*(sizeof(uint32_t)==sizeof(x)) - 1]
00042 #else
00043 #define AAX_ENUM_SIZE_CHECK(x)
00044 #endif
00045
00046
00047 //*****
00048 // ENUM: AAX_EHighlightColor
00049 //*****
00055 enum AAX_EHighlightColor
00056 {
00057     AAX_eHighlightColor_Red = 0,
00058     AAX_eHighlightColor_Blue = 1,
00059     AAX_eHighlightColor_Green = 2,
00060     AAX_eHighlightColor_Yellow = 3,
00061 }
```

```

00062     AAX_eHighlightColor_Num
00063 }; AAX_ENUM_SIZE_CHECK( AAX_EHighlightColor );
00064
00065
00072 enum AAX_ETracePriorityHost
00073 {
00074     AAX_eTracePriorityHost_None = 0,
00075     AAX_eTracePriorityHost_Critical = 0x10000000,
00076     AAX_eTracePriorityHost_High = 0x08000000,
00077     AAX_eTracePriorityHost_Normal = 0x04000000,
00078     AAX_eTracePriorityHost_Low = 0x02000000,
00079     AAX_eTracePriorityHost_Lowest = 0x01000000
00080 }; AAX_ENUM_SIZE_CHECK( AAX_ETracePriorityHost );
00081
00088 enum AAX_ETracePriorityDSP
00089 {
00090     AAX_eTracePriorityDSP_None = 0,
00091     AAX_eTracePriorityDSP_Assert = 1,
00092     AAX_eTracePriorityDSP_High = 2,
00093     AAX_eTracePriorityDSP_Normal = 3,
00094     AAX_eTracePriorityDSP_Low = 4
00095 }; AAX_ENUM_SIZE_CHECK( AAX_ETracePriorityDSP );
00096
00099 enum AAX_EModifiers
00100 {
00101     AAX_eModifiers_None = 0,
00102
00103     AAX_eModifiers_Shift = ( 1 << 0 ),
00104     AAX_eModifiers_Control = ( 1 << 1 ),
00105     AAX_eModifiers_Option = ( 1 << 2 ),
00106     AAX_eModifiers_Command = ( 1 << 3 ),
00107     AAX_eModifiers_SecondaryButton = ( 1 << 4 ),
00108
00109     AAX_eModifiers_Alt = AAX_eModifiers_Option,
00110     AAX_eModifiers_Cntl = AAX_eModifiers_Command,
00111     AAX_eModifiers_WINKEY = AAX_eModifiers_Control
00112 }; AAX_ENUM_SIZE_CHECK( AAX_EModifiers );
00113
00127 enum AAX_EAudioBufferLength
00128 {
00129     AAX_eAudioBufferLength_Undefined = -1,
00130     AAX_eAudioBufferLength_1 = 0,
00131     AAX_eAudioBufferLength_2 = 1,
00132     AAX_eAudioBufferLength_4 = 2,
00133     AAX_eAudioBufferLength_8 = 3,
00134     AAX_eAudioBufferLength_16 = 4,
00135     AAX_eAudioBufferLength_32 = 5,
00136     AAX_eAudioBufferLength_64 = 6,
00137     AAX_eAudioBufferLength_128 = 7,
00138     AAX_eAudioBufferLength_256 = 8,
00139     AAX_eAudioBufferLength_512 = 9,
00140     AAX_eAudioBufferLength_1024 = 10,
00141
00151     AAX_eAudioBufferLength_Max = AAX_eAudioBufferLength_1024
00152 }; AAX_ENUM_SIZE_CHECK( AAX_EAudioBufferLength );
00153
00162 enum AAX_EAudioBufferLengthDSP
00163 {
00164     AAX_eAudioBufferLengthDSP_Default = AAX_eAudioBufferLength_4,
00165     AAX_eAudioBufferLengthDSP_4 = AAX_eAudioBufferLength_4,
00166     AAX_eAudioBufferLengthDSP_16 = AAX_eAudioBufferLength_16,
00167     AAX_eAudioBufferLengthDSP_32 = AAX_eAudioBufferLength_32,
00168     AAX_eAudioBufferLengthDSP_64 = AAX_eAudioBufferLength_64,
00169
00170     AAX_eAudioBufferLengthDSP_Max = AAX_eAudioBufferLengthDSP_64
00171 }; AAX_ENUM_SIZE_CHECK( AAX_EAudioBufferLengthDSP );
00172
00183 enum AAE_EAudioBufferLengthNative
00184 {
00185     AAX_eAudioBufferLengthNative_Min = AAX_eAudioBufferLength_32,
00186     AAX_eAudioBufferLengthNative_Max = AAX_eAudioBufferLength_Max
00187 }; AAX_ENUM_SIZE_CHECK( AAE_EAudioBufferLengthNative );
00188
00194 enum AAX_EMaxAudioSuiteTracks
00195 {
00196     AAX_eMaxAudioSuiteTracks = 48
00197 }; AAX_ENUM_SIZE_CHECK( AAX_EMaxAudioSuiteTracks );
00198
00199 // The channel count ternary here will issue a warning due to a
00200 // signed/unsigned mismatch if anyone tries to create an
00201 // AAX_STEM_FORMAT definition with a negative channel count.
00202 #define AAX_STEM_FORMAT( aIndex, aChannelCount ) ( static_cast<uint32_t>( (
00203     static_cast<uint16_t>(aIndex) << 16 ) | ( (aChannelCount >= AAX_UINT16_MIN) && (aChannelCount <=
00204     0xFFFF) ? aChannelCount & 0xFFFF : 0x0000 ) ) )
00203 #define AAX_STEM_FORMAT_CHANNEL_COUNT( aStemFormat ) ( static_cast<uint16_t>( aStemFormat & 0xFFFF
00204 ) )
00204 #define AAX_STEM_FORMAT_INDEX( aStemFormat ) ( static_cast<int16_t>( ( aStemFormat >> 16 ) &

```

```

    0xFFFF ) )
00205
00229 enum AAX_EStemFormat
00230 {
00231     // Point source stem formats
00232     AAX_eStemFormat_Mono = AAX_STEM_FORMAT ( 0, 1 ),
00233     AAX_eStemFormat_Stereo = AAX_STEM_FORMAT ( 1, 2 ),
00234     AAX_eStemFormat_LCR = AAX_STEM_FORMAT ( 2, 3 ),
00235     AAX_eStemFormat_LCRS = AAX_STEM_FORMAT ( 3, 4 ),
00236     AAX_eStemFormat_Quad = AAX_STEM_FORMAT ( 4, 4 ),
00237     AAX_eStemFormat_5_0 = AAX_STEM_FORMAT ( 5, 5 ),
00238     AAX_eStemFormat_5_1 = AAX_STEM_FORMAT ( 6, 6 ),
00239     AAX_eStemFormat_6_0 = AAX_STEM_FORMAT ( 7, 6 ),
00240     AAX_eStemFormat_6_1 = AAX_STEM_FORMAT ( 8, 7 ),
00241     AAX_eStemFormat_7_0_SDDS = AAX_STEM_FORMAT ( 9, 7 ),
00242     AAX_eStemFormat_7_1_SDDS = AAX_STEM_FORMAT ( 10, 8 ),
00243     AAX_eStemFormat_7_0_DTS = AAX_STEM_FORMAT ( 11, 7 ),
00244     AAX_eStemFormat_7_1_DTS = AAX_STEM_FORMAT ( 12, 8 ),
00245     AAX_eStemFormat_7_0_2 = AAX_STEM_FORMAT ( 20, 9 ),
00246     AAX_eStemFormat_7_1_2 = AAX_STEM_FORMAT ( 13, 10 ),
00247     AAX_eStemFormat_5_0_2 = AAX_STEM_FORMAT ( 21, 7 ),
00248     AAX_eStemFormat_5_1_2 = AAX_STEM_FORMAT ( 22, 8 ),
00249     AAX_eStemFormat_5_0_4 = AAX_STEM_FORMAT ( 23, 9 ),
00250     AAX_eStemFormat_5_1_4 = AAX_STEM_FORMAT ( 24, 10 ),
00251     AAX_eStemFormat_7_0_4 = AAX_STEM_FORMAT ( 25, 11 ),
00252     AAX_eStemFormat_7_1_4 = AAX_STEM_FORMAT ( 26, 12 ),
00253     AAX_eStemFormat_7_0_6 = AAX_STEM_FORMAT ( 35, 13 ),
00254     AAX_eStemFormat_7_1_6 = AAX_STEM_FORMAT ( 36, 14 ),
00255     AAX_eStemFormat_9_0_4 = AAX_STEM_FORMAT ( 27, 13 ),
00256     AAX_eStemFormat_9_1_4 = AAX_STEM_FORMAT ( 28, 14 ),
00257     AAX_eStemFormat_9_0_6 = AAX_STEM_FORMAT ( 29, 15 ),
00258     AAX_eStemFormat_9_1_6 = AAX_STEM_FORMAT ( 30, 16 ),
00259
00260     // Ambisonics stem formats
00261     AAX_eStemFormat_Ambi_1_ACN = AAX_STEM_FORMAT ( 14, 4 ),
00262     AAX_eStemFormat_Ambi_2_ACN = AAX_STEM_FORMAT ( 18, 9 ),
00263     AAX_eStemFormat_Ambi_3_ACN = AAX_STEM_FORMAT ( 19, 16 ),
00264     AAX_eStemFormat_Ambi_4_ACN = AAX_STEM_FORMAT ( 31, 25 ),
00265     AAX_eStemFormat_Ambi_5_ACN = AAX_STEM_FORMAT ( 32, 36 ),
00266     AAX_eStemFormat_Ambi_6_ACN = AAX_STEM_FORMAT ( 33, 49 ),
00267     AAX_eStemFormat_Ambi_7_ACN = AAX_STEM_FORMAT ( 34, 64 ),
00268
00269
00270
00271
00272     AAX_eStemFormatNum = 37, // One greater than the highest available
    AAX_STEM_FORMAT_INDEX value. This needs to increase as stem types are added.
00273
00274     AAX_eStemFormat_None = AAX_STEM_FORMAT ( -100, 0 ),
00275     AAX_eStemFormat_Any = AAX_STEM_FORMAT ( -1, 0 ),
00276
00277     AAX_eStemFormat_INT32_MAX = AAX_INT32_MAX
00278 }; AAX_ENUM_SIZE_CHECK( AAX_EStemFormat );
00279
00295 enum AAX_EPlugInCategory
00296 {
00297     AAX_ePlugInCategory_None = 0x00000000,
00298     AAX_ePlugInCategory_EQ = 0x00000001,
00299     AAX_ePlugInCategory_Dynamics = 0x00000002,
00300     AAX_ePlugInCategory_PitchShift = 0x00000004,
00301     AAX_ePlugInCategory_Reverb = 0x00000008,
00302     AAX_ePlugInCategory_Delay = 0x00000010,
00303     AAX_ePlugInCategory_Modulation = 0x00000020,
00304     AAX_ePlugInCategory_Harmonic = 0x00000040,
00305     AAX_ePlugInCategory_NoiseReduction = 0x00000080,
00306     AAX_ePlugInCategory_Dither = 0x00000100,
00307     AAX_ePlugInCategory_SoundField = 0x00000200,
00308     AAX_ePlugInCategory_HWGGenerators = 0x00000400,
00309     AAX_ePlugInCategory_SWGenerators = 0x00000800,
00310     AAX_ePlugInCategory_WrappedPlugin = 0x00001000,
00311     AAX_EPlugInCategory_Effect = 0x00002000,
00312
00313     // HACK: 32-bit hosts do not have support for AAX_ePlugInCategory_Example
00314     #if ( defined( WIN64 ) || defined( __LP64__ ) )
00315         AAX_ePlugInCategory_Example = 0x00004000,
00316     #else
00317         AAX_ePlugInCategory_Example = AAX_EPlugInCategory_Effect,
00318     #endif
00319
00320     AAX_EPlugInCategory_MIDIEffect = 0x00010000,
00321
00322     AAX_ePlugInCategory_INT32_MAX = AAX_INT32_MAX
00323 }; AAX_ENUM_SIZE_CHECK( AAX_EPlugInCategory );
00324
00334 enum AAX_EPlugInStrings
00335 {
00336     AAX_ePlugInStrings_Analysis = 0,

```

```

00337     AAX_ePlugInStrings_MonoMode = 1,
00338     AAX_ePlugInStrings_MultiInputMode = 2,
00339     AAX_ePlugInStrings_RegionByRegionAnalysis = 3,
00340     AAX_ePlugInStrings_AllSelectedRegionsAnalysis = 4,
00341     AAX_ePlugInStrings_RegionName = 5,
00342     AAX_ePlugInStrings_ClipName = 5,
00343     AAX_ePlugInStrings_Progress = 6,
00344     AAX_ePlugInStrings_PluginFileName = 7,
00345     AAX_ePlugInStrings_Preview = 8,
00346     AAX_ePlugInStrings_Process = 9,
00347     AAX_ePlugInStrings_Bypass = 10,
00348     AAX_ePlugInStrings_ClipNameSuffix = 11,
00349
00350     AAX_ePlugInStrings_INT32_MAX = AAX_INT32_MAX
00351 }; AAX_ENUM_SIZE_CHECK( AAX_EPlugInStrings );
00352
00360 enum AAX_EMeterOrientation
00361 {
00362     AAX_eMeterOrientation_Default = 0,
00363     AAX_eMeterOrientation_BottomLeft = AAX_eMeterOrientation_Default,
00364     AAX_eMeterOrientation_TopRight = 1,
00365     AAX_eMeterOrientation_Center = 2,
00366     AAX_eMeterOrientation_PhaseDot = 3
00367 }; AAX_ENUM_SIZE_CHECK( AAX_EMeterOrientation );
00368
00377 enum AAX_EMeterBallisticType
00378 {
00379     AAX_eMeterBallisticType_Host = 0,
00380     AAX_eMeterBallisticType_NoDecay = 1
00381 }; AAX_ENUM_SIZE_CHECK( AAX_EMeterBallisticType );
00382
00390 enum AAX_EMeterType
00391 {
00392     AAX_eMeterType_Input = 0,
00393     AAX_eMeterType_Output = 1,
00394     AAX_eMeterType_CLGain = 2,
00395     AAX_eMeterType_EGGain = 3,
00396     AAX_eMeterType_Analysis = 4,
00397     AAX_eMeterType_Other = 5,
00398     AAX_eMeterType_None = 31
00399 }; AAX_ENUM_SIZE_CHECK( AAX_EMeterType );
00400
00412 enum AAX_ECurveType
00413 {
00414     AAX_eCurveType_None = 0,
00415
00421     AAX_eCurveType_EQ = 'AXeq',
00427     AAX_eCurveType_Dynamics = 'AXdy',
00433     AAX_eCurveType_Reduction = 'AXdr'
00434 }; AAX_ENUM_SIZE_CHECK( AAX_ECurveType );
00435
00441 enum AAX_EResourceType
00442 {
00443     AAX_eResourceType_None = 0,
00446     AAX_eResourceType_PageTable,
00451     AAX_eResourceType_PageTableDir
00452 }; AAX_ENUM_SIZE_CHECK( AAX_EResourceType );
00453
00469 enum AAX_ENotificationEvent
00470 {
00477     AAX_eNotificationEvent_InsertPositionChanged = 'AXip',
00487     AAX_eNotificationEvent_TrackNameChanged = 'AXtn',
00494     AAX_eNotificationEvent_TrackUIDChanged = 'AXtu',
00501     AAX_eNotificationEvent_TrackPositionChanged = 'AXtp',
00507     AAX_eNotificationEvent_AlgorithmMoved = 'AXam',
00513     AAX_eNotificationEvent_GUIOpened = 'AXgo',
00519     AAX_eNotificationEvent_GUIClosed = 'AXgc',
00529     AAX_eNotificationEvent_ASProcessingState = 'AXpr',
00539     AAX_eNotificationEvent_ASPreviewState = 'ASpv',
00548     AAX_eNotificationEvent_SessionBeingOpened = 'AXso',
00556     AAX_eNotificationEvent_PresetOpened = 'AXpo',
00564     AAX_eNotificationEvent_EnteringOfflineMode = 'AXof',
00572     AAX_eNotificationEvent_ExitingOfflineMode = 'AXox',
00581     AAX_eNotificationEvent_SessionPathChanged = 'AXsp',
00594     AAX_eNotificationEvent_SignalLatencyChanged = 'AXsl',
00613     AAX_eNotificationEvent_DelayCompensationState = 'AXdc',
00620     AAX_eNotificationEvent_CycleCountChanged = 'AXcc',
00628     AAX_eNotificationEvent_MaxViewSizeChanged = 'AXws',
00636     AAX_eNotificationEvent_SideChainBeingConnected = 'AXsc',
00645     AAX_eNotificationEvent_SideChainBeingDisconnected = 'AXsd',
00659     AAX_eNotificationEvent_NoiseFloorChanged = 'AXnf',
00668     AAX_eNotificationEvent_ParameterMappingChanged = 'AXpm',
00680     AAX_eNotificationEvent_ParameterNameChanged = 'AXpn',
00688     AAX_eNotificationEvent_HostModeChanged = 'AXhm',
00708     AAX_eNotificationEvent_PriorSettingsInvalid = 'AXps',
00719     AAX_eNotificationEvent_LogState = 'AXls',

```

```

00727     AAX_eNotificationEvent_TransportStateChanged = 'AXts',
00728 }; AAX_ENUM_SIZE_CHECK( AAX_ENotificationEvent );
00729
00730
00731 enum AAX_EHostModeBits
00732 {
00733     AAX_eHostModeBits_None = 0,
00734     AAX_eHostModeBits_Live = (1 << 0)
00735 }; AAX_ENUM_SIZE_CHECK( AAX_EHostModeBits );
00736
00737 enum AAX_EHostMode
00738 {
00739     AAX_eHostMode_Show = AAX_eHostModeBits_Live,
00740     AAX_eHostMode_Config = AAX_eHostModeBits_None
00741 }; AAX_ENUM_SIZE_CHECK( AAX_EHostMode );
00742
00743 enum AAX_EPrivateDataOptions
00744 {
00745     AAX_ePrivateDataOptions_DefaultOptions = 0,
00746     AAX_ePrivateDataOptions_KeepOnReset = (1 << 0),
00747     AAX_ePrivateDataOptions_External = (1 << 1),
00748     AAX_ePrivateDataOptions_Align8 = (1 << 2),
00749     AAX_ePrivateDataOptions_INT32_MAX = AAX_INT32_MAX
00750 }; AAX_ENUM_SIZE_CHECK( AAX_EPrivateDataOptions );
00751
00752 enum AAX_EConstraintLocationMask
00753 {
00754     AAX_eConstraintLocationMask_None = 0,
00755     AAX_eConstraintLocationMask_DataModel = (1 << 0),
00756     AAX_eConstraintLocationMask_DLLChipAffinity = (1 << 1),
00757 }; AAX_ENUM_SIZE_CHECK( AAX_EConstraintLocationMask );
00758
00759 enum AAX_EConstraintTopology
00760 {
00761     AAX_eConstraintTopology_None = 0,
00762     AAX_eConstraintTopology_Monolithic = 1
00763 }; AAX_ENUM_SIZE_CHECK( AAX_EConstraintTopology );
00764
00765 enum AAX_EComponentInstanceInitAction
00766 {
00767     AAX_eComponentInstanceInitAction_AddingNewInstance = 0,
00768     AAX_eComponentInstanceInitAction_RemovingInstance = 1,
00769     AAX_eComponentInstanceInitAction_ResetInstance = 2
00770 }; AAX_ENUM_SIZE_CHECK( AAX_EComponentInstanceInitAction );
00771
00772 enum AAX_ESampleRateMask
00773 {
00774     AAX_eSampleRateMask_No = 0,
00775     AAX_eSampleRateMask_44100 = (1 << 0),
00776     AAX_eSampleRateMask_48000 = (1 << 1),
00777     AAX_eSampleRateMask_88200 = (1 << 2),
00778     AAX_eSampleRateMask_96000 = (1 << 3),
00779     AAX_eSampleRateMask_176400 = (1 << 4),
00780     AAX_eSampleRateMask_192000 = (1 << 5),
00781     AAX_eSampleRateMask_All = AAX_INT32_MAX
00782 }; AAX_ENUM_SIZE_CHECK( AAX_ESampleRateMask );
00783
00784 typedef enum AAX_EParameterType
00785 {
00786     AAX_eParameterType_Discrete,
00787     AAX_eParameterType_Continuous
00788 } AAX_EParameterType; AAX_ENUM_SIZE_CHECK( AAX_EParameterType );
00789
00790 enum AAX_EParameterOrientationBits {
00791     AAX_eParameterOrientation_Default = 0,
00792     AAX_eParameterOrientation_BottomMinTopMax = 0,           // Choose this...
00793     AAX_eParameterOrientation_TopMinBottomMax = 1,           // or this.
00794     AAX_eParameterOrientation_LeftMinRightMax = 0,           // AND this...
00795     AAX_eParameterOrientation_RightMinLeftMax = 2,           // or this.
00796     // Rotary multi-Segment Display Choices
00797     AAX_eParameterOrientation_RotarySingleDotMode = 0,       // AND this...
00798     AAX_eParameterOrientation_RotaryBoostCutMode = 4,         // or this.
00799     AAX_eParameterOrientation_RotaryWrapMode = 8,             // or this.
00800     AAX_eParameterOrientation_RotarySpreadMode = 12,          // or this.
00801     // Rotary multi-Segment Display Polarity
00802     AAX_eParameterOrientation_RotaryLeftMinRightMax = 0,     // AND this...

```



```

00880     AAX_eParameterOrientation_RotaryRightMinLeftMax = 16    // or this.
00881 }; AAX_ENUM_SIZE_CHECK( AAX_EParameterOrientationBits );
00882
00885 typedef int32_t      AAX_EParameterOrientation;
00886
00896 enum AAX_EParameterValueInfoSelector
00897 {
00906     AAX_ePageTable_EQ_Band_Type = 0,
00915     AAX_ePageTable_EQ_InCircuitPolarity = 1,
00933     AAX_ePageTable_UseAlternateControl = 2,
00934 }; AAX_ENUM_SIZE_CHECK( AAX_EParameterValueInfoSelector );
00935
00941 enum AAX_EEQBandTypes
00942 {
00943     AAX_eEQBandType_HighPass = 0,
00944     AAX_eEQBandType_LowShelf = 1,
00945     AAX_eEQBandType_Parametric = 2,
00946     AAX_eEQBandType_HighShelf = 3,
00947     AAX_eEQBandType_LowPass = 4,
00948     AAX_eEQBandType_Notch = 5
00949 }; AAX_ENUM_SIZE_CHECK( AAX_EEQBandTypes );
00950
00956 enum AAX_EEQInCircuitPolarity
00957 {
00958     AAX_eEQInCircuitPolarity_Enabled = 0,
00959     AAX_eEQInCircuitPolarity_Bypassed = 1,
00960     AAX_eEQInCircuitPolarity_Disabled = 2
00961 }; AAX_ENUM_SIZE_CHECK( AAX_EEQInCircuitPolarity );
00962
00968 enum AAX_EUseAlternateControl
00969 {
00970     AAX_eUseAlternateControl_No = 0,
00971     AAX_eUseAlternateControl_Yes = 1
00972 }; AAX_ENUM_SIZE_CHECK( AAX_EUseAlternateControl );
00973
00979 enum AAX_EMIDINodeType
00980 {
00991     AAX_eMIDINodeType_LocalInput = 0,
01011     AAX_eMIDINodeType_LocalOutput = 1,
01033     AAX_eMIDINodeType_Global = 2,
01041     AAX_eMIDINodeType_Transport = 3
01042 }; AAX_ENUM_SIZE_CHECK( AAX_EMIDINodeType );
01043
01044
01048 enum AAX_EUpdateSource
01049 {
01050     AAX_eUpdateSource_Unspecified = 0,
01051     AAX_eUpdateSource_Parameter = 1,
01052     AAX_eUpdateSource_Chunk = 2,
01053     AAX_eUpdateSource_Delay = 3
01054 }; AAX_ENUM_SIZE_CHECK( AAX_EUpdateSource );
01055
01062 enum AAX_EDataInPortType
01063 {
01066     AAX_eDataInPortType_Unbuffered = 0,
01073     AAX_eDataInPortType_Buffered = 1,
01086     AAX_eDataInPortType_Incremental = 2
01087 }; AAX_ENUM_SIZE_CHECK( AAX_EDataInPortType );
01088
01095 enum AAX_EFrameRate
01096 {
01097     AAX_eFrameRate_Undeclared = 0,
01098     AAX_eFrameRate_24Frame = 1,
01099     AAX_eFrameRate_25Frame = 2,
01100     AAX_eFrameRate_2997NonDrop = 3,
01101     AAX_eFrameRate_2997DropFrame = 4,
01102     AAX_eFrameRate_30NonDrop = 5,
01103     AAX_eFrameRate_30DropFrame = 6,
01104     AAX_eFrameRate_23976 = 7,
01105     AAX_eFrameRate_47952 = 8,
01106     AAX_eFrameRate_48Frame = 9,
01107     AAX_eFrameRate_50Frame = 10,
01108     AAX_eFrameRate_5994NonDrop = 11,
01109     AAX_eFrameRate_5994DropFrame = 12,
01110     AAX_eFrameRate_60NonDrop = 13,
01111     AAX_eFrameRate_60DropFrame = 14,
01112     AAX_eFrameRate_100Frame = 15,
01113     AAX_eFrameRate_11988NonDrop = 16,
01114     AAX_eFrameRate_11988DropFrame = 17,
01115     AAX_eFrameRate_120NonDrop = 18,
01116     AAX_eFrameRate_120DropFrame = 19
01117 }; AAX_ENUM_SIZE_CHECK( AAX_EFrameRate );
01118
01124 enum AAX_EFeetFramesRate
01125 {
01126     AAX_eFeetFramesRate_23976 = 0,
01127     AAX_eFeetFramesRate_24 = 1,

```

```

01128     AAX_eFeetFramesRate_25 = 2
01129 }; AAX_ENUM_SIZE_CHECK( AAX_EFeetFramesRate );
01130
01131
01132 enum AAX_EMidiglobalNodeSelectors
01133 {
01134     AAX_eMIDIClick          = 1 << 0,
01135     AAX_eMIDIMtc           = 1 << 1,
01136     AAX_eMIDIBeatClock     = 1 << 2
01137 }; AAX_ENUM_SIZE_CHECK( AAX_EMidiglobalNodeSelectors );
01138
01139 enum AAX_EPreviewState
01140 {
01141     AAX_ePreviewState_Stop = 0,
01142     AAX_ePreviewState_Start = 1
01143 }; AAX_ENUM_SIZE_CHECK( AAX_EPreviewState );
01144
01145 enum AAX_EProcessingState
01146 {
01147     AAX_eProcessingState_StopPass = 2,
01148     AAX_eProcessingState_StartPass = 3,
01149     AAX_eProcessingState_EndPassGroup = 4,
01150     AAX_eProcessingState_BeginPassGroup = 5,
01151
01152     AAX_eProcessingState_Stop = AAX_eProcessingState_StopPass,
01153     AAX_eProcessingState_Start = AAX_eProcessingState_StartPass
01154 }; AAX_ENUM_SIZE_CHECK( AAX_EProcessingState );
01155
01156 enum AAX_ETargetPlatform
01157 {
01158     kAAX_eTargetPlatform_None = 0,
01159     kAAX_eTargetPlatform_Native = 1,      // For host-based components
01160     kAAX_eTargetPlatform_TI = 2,         // For TI components
01161     kAAX_eTargetPlatform_External = 3,    // For components running on external hardware
01162     kAAX_eTargetPlatform_Count = 5
01163 }; AAX_ENUM_SIZE_CHECK( AAX_ETargetPlatform );
01164
01165 enum AAX_ESupportLevel
01166 {
01167     AAX_eSupportLevel_Uninitialized = 0,
01168
01169     AAX_eSupportLevel_Unsupported = 1
01170     , AAX_eSupportLevel_Supported = 2
01171
01172     , AAX_eSupportLevel_Disabled = 3
01173
01174     , AAX_eSupportLevel_ByProperty = 4
01175 }; AAX_ENUM_SIZE_CHECK( AAX_ESupportLevel );
01176
01177 enum AAX_EHostLevel
01178 {
01179     AAX_eHostLevel_Unknown = 0
01180     , AAX_eHostLevel_Standard = 1
01181     , AAX_eHostLevel_Entry = 2
01182     , AAX_eHostLevel_Intermediate = 3
01183 }; AAX_ENUM_SIZE_CHECK( AAX_EHostLevel );
01184
01185 enum AAX_ETextEncoding
01186 {
01187     AAX_eTextEncoding_Undefined = -1
01188     , AAX_eTextEncoding_UTF8 = 0
01189
01190     , AAX_eTextEncoding_Num
01191 }; AAX_ENUM_SIZE_CHECK( AAX_ETextEncoding );
01192
01193 enum AAX_EAssertFlags
01194 {
01195     AAX_eAssertFlags_Default = 0,
01196     AAX_eAssertFlags_Log = 1 << 0,
01197     AAX_eAssertFlags_Dialog = 1 << 1,
01198 }; AAX_ENUM_SIZE_CHECK( AAX_EAssertFlags );
01199
01200 // ENUM: AAX_ETransportState
01201 enum AAX_ETransportState
01202 {
01203     AAX_eTransportState_Unknown = 0,
01204     AAX_eTransportState_Stopping = 1,
01205     AAX_eTransportState_Stop = 2,
01206     AAX_eTransportState_Paused = 3,
01207     AAX_eTransportState_Play = 4,
01208     AAX_eTransportState_FastForward = 5,
01209     AAX_eTransportState_Rewind = 6,
01210     AAX_eTransportState_Scrub = 11,
01211     AAX_eTransportState_Shuttle = 12,
01212

```

```

01363     AAX_eTransportState_Num
01364 }; AAX_ENUM_SIZE_CHECK(AAX_ETransportState);
01365
01366 // ENUM: AAX_ERecordMode
01370 enum AAX_ERecordMode
01371 {
01372     AAX_eRecordMode_Unknown = 0,
01373     AAX_eRecordMode_None = 1,
01374     AAX_eRecordMode_Normal = 2,
01375     AAX_eRecordMode_Destructive = 3,
01376     AAX_eRecordMode_QuickPunch = 4,
01377     AAX_eRecordMode_TrackPunch = 5,
01378
01379     AAX_eRecordMode_Num
01380 }; AAX_ENUM_SIZE_CHECK(AAX_ERecordMode);
01381
01383 #endif // include guard

```

15.129 AAX_EnvironmentUtilities.h File Reference

```
#include <cstdlib>
```

15.129.1 Description

Useful environment definitions for AAX.

Namespaces

- namespace [AAX](#)

15.130 AAX_EnvironmentUtilities.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003
00004     AAX_EnvironmentUtilities.h
00005
00006     Copyright 2018–2019, 2023 Avid Technology, Inc.
00007     All rights reserved.
00008
00009     CONFIDENTIAL: this document contains confidential information of Avid. Do
00010     not disclose to any third party. Use of the information contained in this
00011     document is subject to an Avid SDK license.
00012 */
00013
00020 /*=====*/
00021
00022 #ifndef _AAX_ENVIRONMENTUTILITIES_H_
00023 #define _AAX_ENVIRONMENTUTILITIES_H_
00024
00025 #include <cstdlib>
00026
00027 #if (!defined (WINDOWS_VERSION))
00028 #   if (defined (__WIN32))
00029 #       define WINDOWS_VERSION 1
00030 #   endif
00031 #elif (defined (MAC_VERSION) || defined (LINUX_VERSION))
00032 #   error "AAX SDK: Cannot declare more than one OS environment"
00033 #endif
00034
00035 #if (!defined (MAC_VERSION))
00036 #   if (defined (__APPLE__) && defined (__MACH__))
00037 #       include "TargetConditionals.h"
00038 #       if (TARGET_OS_MAC)

```

```

00039 #         define MAC_VERSION 1
00040 #     endif
00041 # endif
00042 #elif (defined (WINDOWS_VERSION) || defined (LINUX_VERSION))
00043 #     error "AAX SDK: Cannot declare more than one OS environment"
00044 #endif
00045
00046 #if (!defined (LINUX_VERSION))
00047 #     if (defined (__linux__))
00048 #         define LINUX_VERSION 1
00049 #     endif
00050 #elif (defined (WINDOWS_VERSION) || defined (MAC_VERSION))
00051 #     error "AAX SDK: Cannot declare more than one OS environment"
00052 #endif
00053
00054 #if (!defined (WINDOWS_VERSION) && !defined (MAC_VERSION) && !defined (LINUX_VERSION))
00055 #     warning "AAX SDK: Unknown OS environment"
00056 #endif
00057
00058 namespace AAX
00059 {
00060     static bool IsVenueSystem(void)
00061     {
00062         #if WINDOWS_VERSION
00063             static const char * const environmentVariableName = "JEX_HOST_TYPE";
00064             static const char * const venueEnvironment = "venue";
00065             static const char * const environment = std::getenv ( environmentVariableName );
00066             static const bool isVenue = ( NULL != environment) && (0 == strcmp ( environment,
venueEnvironment ) );
00067             return isVenue;
00068         #else
00069             return false;
00070         #endif
00071     }
00072 }
00073
00074 #endif // _AAX_ENVIRONMENTUTILITIES_H_

```

15.131 AAX_Errors.h File Reference

```
#include "AAX_Enums.h"
```

15.131.1 Description

Definitions of error codes used by AAX plug-ins.

Enumerations

- enum [AAX_EError](#) {
 - [AAX_SUCCESS](#) = 0 ,
 - [AAX_ERROR_INVALID_PARAMETER_ID](#) = -20001 ,
 - [AAX_ERROR_INVALID_STRING_CONVERSION](#) = -20002 ,
 - [AAX_ERROR_INVALID_METER_INDEX](#) = -20003 ,
 - [AAX_ERROR_NULL_OBJECT](#) = -20004 ,
 - [AAX_ERROR_OLDER_VERSION](#) = -20005 ,
 - [AAX_ERROR_INVALID_CHUNK_INDEX](#) = -20006 ,
 - [AAX_ERROR_INVALID_CHUNK_ID](#) = -20007 ,
 - [AAX_ERROR_INCORRECT_CHUNK_SIZE](#) = -20008 ,
 - [AAX_ERROR_UNIMPLEMENTED](#) = -20009 ,
 - [AAX_ERROR_INVALID_PARAMETER_INDEX](#) = -20010 ,
 - [AAX_ERROR_NOT_INITIALIZED](#) = -20011 ,
 - [AAX_ERROR_ACF_ERROR](#) = -20012 ,
 - [AAX_ERROR_INVALID_METER_TYPE](#) = -20013 ,

```

AAX_ERROR_CONTEXT_ALREADY_HAS_METERS = -20014 ,
AAX_ERROR_NULL_COMPONENT = -20015 ,
AAX_ERROR_PORT_ID_OUT_OF_RANGE = -20016 ,
AAX_ERROR_FIELD_TYPE_DOES_NOT_SUPPORT_DIRECT_ACCESS = -20017 ,
AAX_ERROR_DIRECT_ACCESS_OUT_OF_BOUNDS = -20018 ,
AAX_ERROR_FIFO_FULL = -20019 ,
AAX_ERROR_INITIALIZING_PACKET_STREAM_THREAD = -20020 ,
AAX_ERROR_POST_PACKET_FAILED = -20021 ,
AAX_RESULT_PACKET_STREAM_NOT_EMPTY = -20022 ,
AAX_RESULT_ADD_FIELD_UNSUPPORTED_FIELD_TYPE = -20023 ,
AAX_ERROR_MIXER_THREAD_FALLING_BEHIND = -20024 ,
AAX_ERROR_INVALID_FIELD_INDEX = -20025 ,
AAX_ERROR_MALFORMED_CHUNK = -20026 ,
AAX_ERROR_TOD_BEHIND = -20027 ,
AAX_RESULT_NEW_PACKET_POSTED = -20028 ,
AAX_ERROR_PLUGIN_NOT_AUTHORIZED = -20029 ,
AAX_ERROR_PLUGIN_NULL_PARAMETER = -20030 ,
AAX_ERROR_NOTIFICATION_FAILED = -20031 ,
AAX_ERROR_INVALID_VIEW_SIZE = -20032 ,
AAX_ERROR_SIGNED_INT_OVERFLOW = -20033 ,
AAX_ERROR_NO_COMPONENTS = -20034 ,
AAX_ERROR_DUPLICATE_EFFECT_ID = -20035 ,
AAX_ERROR_DUPLICATE_TYPE_ID = -20036 ,
AAX_ERROR_EMPTY_EFFECT_NAME = -20037 ,
AAX_ERROR_UNKNOWN_PLUGIN = -20038 ,
AAX_ERROR_PROPERTY_UNDEFINED = -20039 ,
AAX_ERROR_INVALID_PATH = -20040 ,
AAX_ERROR_UNKNOWN_ID = -20041 ,
AAX_ERROR_UNKNOWN_EXCEPTION = -20042 ,
AAX_ERROR_INVALID_ARGUMENT = -20043 ,
AAX_ERROR_NULL_ARGUMENT = -20044 ,
AAX_ERROR_INVALID_INTERNAL_DATA = -20045 ,
AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW = -20046 ,
AAX_ERROR_UNSUPPORTED_ENCODING = -20047 ,
AAX_ERROR_UNEXPECTED_EFFECT_ID = -20048 ,
AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME = -20049 ,
AAX_ERROR_ARGUMENT_OUT_OF_RANGE = -20050 ,
AAX_ERROR_PRINT_FAILURE = -20051 ,
AAX_ERROR_PLUGIN_BEGIN = -20600 ,
AAX_ERROR_PLUGIN_END = -21000 }

```

Functions

- [AAX_ENUM_SIZE_CHECK](#) ([AAX_EError](#))

15.131.2 Enumeration Type Documentation

15.131.2.1 AAX_EError

enum [AAX_EError](#)

[AAX](#) result codes

Enumerator

AAX_SUCCESS	
AAX_ERROR_INVALID_PARAMETER_ID	
AAX_ERROR_INVALID_STRING_CONVERSION	
AAX_ERROR_INVALID_METER_INDEX	
AAX_ERROR_NULL_OBJECT	
AAX_ERROR_OLDER_VERSION	
AAX_ERROR_INVALID_CHUNK_INDEX	
AAX_ERROR_INVALID_CHUNK_ID	
AAX_ERROR_INCORRECT_CHUNK_SIZE	
AAX_ERROR_UNIMPLEMENTED	
AAX_ERROR_INVALID_PARAMETER_INDEX	
AAX_ERROR_NOT_INITIALIZED	
AAX_ERROR_ACF_ERROR	
AAX_ERROR_INVALID_METER_TYPE	
AAX_ERROR_CONTEXT_ALREADY_HAS_↵ METERS	
AAX_ERROR_NULL_COMPONENT	
AAX_ERROR_PORT_ID_OUT_OF_RANGE	
AAX_ERROR_FIELD_TYPE_DOES_NOT_↵ SUPPORT_DIRECT_ACCESS	
AAX_ERROR_DIRECT_ACCESS_OUT_OF_↵ BOUNDS	
AAX_ERROR_FIFO_FULL	
AAX_ERROR_INITIALIZING_PACKET_STREAM_↵ THREAD	
AAX_ERROR_POST_PACKET_FAILED	
AAX_RESULT_PACKET_STREAM_NOT_EMPTY	
AAX_RESULT_ADD_FIELD_UNSUPPORTED_↵ FIELD_TYPE	
AAX_ERROR_MIXER_THREAD_FALLING_BEHIND	
AAX_ERROR_INVALID_FIELD_INDEX	
AAX_ERROR_MALFORMED_CHUNK	
AAX_ERROR_TOD_BEHIND	
AAX_RESULT_NEW_PACKET_POSTED	
AAX_ERROR_PLUGIN_NOT_AUTHORIZED	
AAX_ERROR_PLUGIN_NULL_PARAMETER	
AAX_ERROR_NOTIFICATION_FAILED	
AAX_ERROR_INVALID_VIEW_SIZE	
AAX_ERROR_SIGNED_INT_OVERFLOW	
AAX_ERROR_NO_COMPONENTS	
AAX_ERROR_DUPLICATE_EFFECT_ID	
AAX_ERROR_DUPLICATE_TYPE_ID	
AAX_ERROR_EMPTY_EFFECT_NAME	
AAX_ERROR_UNKNOWN_PLUGIN	
AAX_ERROR_PROPERTY_UNDEFINED	
AAX_ERROR_INVALID_PATH	
AAX_ERROR_UNKNOWN_ID	
AAX_ERROR_UNKNOWN_EXCEPTION	An AAX plug-in should return this to the host if an unknown exception is caught. Exceptions should never be passed to the host.

Enumerator

AAX_ERROR_INVALID_ARGUMENT	One or more input parameters are invalid; all output parameters are left unchanged.
AAX_ERROR_NULL_ARGUMENT	One or more required pointer arguments are null.
AAX_ERROR_INVALID_INTERNAL_DATA	Some part of the internal data required by the method is invalid. See also AAX_ERROR_NOT_INITIALIZED
AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW	A buffer argument was not large enough to hold the data which must be placed within it.
AAX_ERROR_UNSUPPORTED_ENCODING	Unsupported input argument text encoding.
AAX_ERROR_UNEXPECTED_EFFECT_ID	Encountered an effect ID with a different value from what was expected.
AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME	No parameter name abbreviation with the requested properties has been defined.
AAX_ERROR_ARGUMENT_OUT_OF_RANGE	One or more input parameters are out of the expected range, e.g. an index argument that is negative or exceeds the number of elements.
AAX_ERROR_PRINT_FAILURE	A failure occurred in a "print" library call such as <code>printf</code> .
AAX_ERROR_PLUGIN_BEGIN	Custom plug-in error codes may be placed in the range (AAX_ERROR_PLUGIN_END , AAX_ERROR_PLUGIN_BEGIN].
AAX_ERROR_PLUGIN_END	Custom plug-in error codes may be placed in the range (AAX_ERROR_PLUGIN_END , AAX_ERROR_PLUGIN_BEGIN].

15.131.3 Function Documentation

15.131.3.1 AAX_ENUM_SIZE_CHECK()

```
AAX_ENUM_SIZE_CHECK (
    AAX_EError )
```

15.132 AAX_Errors.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2010-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
```

```

00012
00019 /*=====*/
00020
00021
00023 #ifndef AAX_ERRORS_H
00024 #define AAX_ERRORS_H
00026
00027 #include "AAX_Enums.h"
00028
00034 enum AAX_EError
00035 {
00036     AAX_SUCCESS = 0,
00037
00038     AAX_ERROR_INVALID_PARAMETER_ID = -20001,
00039     AAX_ERROR_INVALID_STRING_CONVERSION = -20002,
00040     AAX_ERROR_INVALID_METER_INDEX = -20003,
00041     AAX_ERROR_NULL_OBJECT = -20004,
00042     AAX_ERROR_OLDER_VERSION = -20005,
00043     AAX_ERROR_INVALID_CHUNK_INDEX = -20006,
00044     AAX_ERROR_INVALID_CHUNK_ID = -20007,
00045     AAX_ERROR_INCORRECT_CHUNK_SIZE = -20008,
00046     AAX_ERROR_UNIMPLEMENTED = -20009,
00047     AAX_ERROR_INVALID_PARAMETER_INDEX = -20010,
00048     AAX_ERROR_NOT_INITIALIZED = -20011,
00049     AAX_ERROR_ACF_ERROR = -20012,
00050     AAX_ERROR_INVALID_METER_TYPE = -20013,
00051     AAX_ERROR_CONTEXT_ALREADY_HAS_METERS = -20014,
00052     AAX_ERROR_NULL_COMPONENT = -20015,
00053     AAX_ERROR_PORT_ID_OUT_OF_RANGE = -20016,
00054     AAX_ERROR_FIELD_TYPE_DOES_NOT_SUPPORT_DIRECT_ACCESS = -20017,
00055     AAX_ERROR_DIRECT_ACCESS_OUT_OF_BOUNDS = -20018,
00056     AAX_ERROR_FIFO_FULL = -20019,
00057     AAX_ERROR_INITIALIZING_PACKET_STREAM_THREAD = -20020,
00058     AAX_ERROR_POST_PACKET_FAILED = -20021,
00059     AAX_RESULT_PACKET_STREAM_NOT_EMPTY = -20022,
00060     AAX_RESULT_ADD_FIELD_UNSUPPORTED_FIELD_TYPE = -20023,
00061     AAX_ERROR_MIXER_THREAD_FALLING_BEHIND = -20024,
00062     AAX_ERROR_INVALID_FIELD_INDEX = -20025,
00063     AAX_ERROR_MALFORMED_CHUNK = -20026,
00064     AAX_ERROR_TOD_BEHIND = -20027,
00065     AAX_RESULT_NEW_PACKET_POSTED = -20028,
00066     AAX_ERROR_PLUGIN_NOT_AUTHORIZED = -20029, //return this from
    EffectInit() if the plug-in doesn't have proper license.
00067     AAX_ERROR_PLUGIN_NULL_PARAMETER = -20030,
00068     AAX_ERROR_NOTIFICATION_FAILED = -20031,
00069     AAX_ERROR_INVALID_VIEW_SIZE = -20032,
00070     AAX_ERROR_SIGNED_INT_OVERFLOW = -20033,
00071     AAX_ERROR_NO_COMPONENTS = -20034,
00072     AAX_ERROR_DUPLICATE_EFFECT_ID = -20035,
00073     AAX_ERROR_DUPLICATE_TYPE_ID = -20036,
00074     AAX_ERROR_EMPTY_EFFECT_NAME = -20037,
00075     AAX_ERROR_UNKNOWN_PLUGIN = -20038,
00076     AAX_ERROR_PROPERTY_UNDEFINED = -20039,
00077     AAX_ERROR_INVALID_PATH = -20040,
00078     AAX_ERROR_UNKNOWN_ID = -20041,
00079     AAX_ERROR_UNKNOWN_EXCEPTION = -20042,
00080     AAX_ERROR_INVALID_ARGUMENT = -20043,
00081     AAX_ERROR_NULL_ARGUMENT = -20044,
00082     AAX_ERROR_INVALID_INTERNAL_DATA = -20045,
00083     AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW = -20046,
00084     AAX_ERROR_UNSUPPORTED_ENCODING = -20047,
00085     AAX_ERROR_UNEXPECTED_EFFECT_ID = -20048,
00086     AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME = -20049,
00087     AAX_ERROR_ARGUMENT_OUT_OF_RANGE = -20050,
00088     AAX_ERROR_PRINT_FAILURE = -20051,
00089
00090
00091     AAX_ERROR_PLUGIN_BEGIN = -20600,
00092     AAX_ERROR_PLUGIN_END = -21000
00093 }; AAX_ENUM_SIZE_CHECK( AAX_EError );
00094
00095
00096
00097
00098
00099
00100
00101
00102
00105 // AAE and other known AAX host error codes //
00106 // Listed here as a reference //
00109
00110 // FicErrors.h
00111 /*
00112
00113 //
00114 // NOTE: (Undefined) comments for an error code mean that it's

```



```

00115 // either no longer supported or returned from another source
00116 // other than DAE.
00117 //
00118
00119 //-----
00120 // Error codes for all of Fic
00121 //-----
00122
00123 enum {
00124     kFicHostTimeoutErr          = -9003,    // Host Timeout Error.  DSP is not responding.
00125     kFicHostBusyErr             = -9004,    // (Undefined)
00126     kFicLowMemoryErr            = -9006,    // DAE was unable to allocate memory.  Memory is low.
00127     kFicUnimplementedErr        = -9007,    // An unimplemented method was called.
00128     kFicAllocatedErr            = -9008,    // (Undefined)
00129     kFicNilObjectErr            = -9013,    // Standard error return when an object is NULL.
00130     kFicNoDriverDSPErr          = -9014,    // Missing DSPPtr from the SADriver.
00131     kFicBadIndexErr             = -9015,    // Index to an array or list is invalid.
00132     kFicAlreadyDeferredErr      = -9017,    // Tried to install a deferred task when the task was
already deferred.
00133     kFicFileSystemBusyErr       = -9019,    // PB chain for an audio file returned an error for a
disk task.
00134     kFicRunningErr              = -9020,    // Tried to execute code when the deck was started.
00135     kFicTooManyItemsErr         = -9022,    // Number of needed items goes beyond a lists max
size.
00136     kFicItemNotFoundErr         = -9023,    // Unable to find an object in a list of objects.
00137     kFicWrongTypeErr            = -9024,    // Type value not found or not supported.
00138     kFicNoDeckErr               = -9025,    // Standard error returned from other objects that
require a deck object.
00139     kFicNoDSPErr                = -9028,    // Required DSP object is NULL.
00140     kFicNoFeederErr             = -9029,    // (Undefined)
00141     kFicNoOwnerErr              = -9030,    // Play or record track not owned by a channel.
00142     kFicPrimedErr               = -9031,    // Tried to execute code when the deck was primed.
00143     kFicAlreadyAttached         = -9032,    // DAE object already attached to another DAE object.
00144     kFicTooManyDSPTracksErr     = -9033,    // The user has run out of virtual tracks for a given
card or dsp.
00145     kFicParseErr                = -9035,    // While trying to parse a data structure ran into an
error.
00146     kFicNotAcquiredErr          = -9041,    // Tried to execute code when an object needs to be
acquired first.
00147     kFicNoSSIClockErr           = -9045,    // DSP does not receive peripheral clock interrupts.
00148     kFicNotFound                = -9048,    // Missing DAE resource or timeout occurred while
waiting for DAE to launch.
00149     kFicCantRecordErr           = -9050,    // Error returned when CanRecord() returns false.
Exp: Recording on scrub channel.
00150     kFicWrongObjectErr          = -9054,    // Object size or pointers do not match.
00151     kFicLowVersionErr           = -9055,    // Errors with version number too low.
00152     kFicNotStartedErr           = -9057,    // Tried to execute code when the deck was not started
yet.
00153     kFicOnly1PunchInErr         = -9059,    // Error when deck can only support a single punch in.
00154     kFicAssertErr               = -9060,    // Generic error when a format does not match.
00155     kFicScrubOnlyErr            = -9061,    // Tried to scrub in a non-scrub mode or on a sys axe
channel.
00156     kFicNoSADriverErr           = -9062,    // InitSADriver failed.  Possible missing DigiSystem
INIT.
00157     kFicCantFindDAEFolder       = -9064,    // Unable to find "DAE Folder" in the system folder.
00158     kFicCantFindDAEApp          = -9065,    // Unable to find DAE app in the DAE Folder.
00159     kFicNeeds32BitModeErr        = -9066,    // DAE runs only in 32 bit mode.
00160     kFicHatesVirtualMemErr       = -9068,    // DAE will not run if virtual memory is turned on.
00161     kFicSCIConnectErr           = -9070,    // Unable to get SCI ports between two dsp's to
communicate.
00162     kFicSADriverVersionErr       = -9071,    // Unable to get DigiSystem INIT version or it's
version is too low.
00163     kFicUserCancelledErr         = -9072,    // User chose to cancel or quit operation from DAE
dialog.
00164     kFicDiskTooSlowErr          = -9073,    // Disk action did not complete in time for next
command.
00165     kFicAudioTrackTooDense1     = -9074,    // Audio playlist is too dense.
00166     kFicAudioTrackTooDense2     = -9075,    // Audio playlist is too dense for silence play list.
00167     kFicCantDescribeZone        = -9076,    // Zone description is NULL.
00168     kFicCantApplyPlayLimits     = -9077,    // Ran out of time regions for a zone.
00169     kFicCantApplySkipMode       = -9078,    // Ran out of time regions for a zone in skip mode.
00170     kFicCantApplyLoop           = -9079,    // Ran out of time regions for a zone in loop mode.
00171     kFicAutoSortErr             = -9084,    // DSP event elements are not sorted in relation to
time.
00172     kFicNoAutoEvent             = -9085,    // No event list for an auto parser.
00173     kFicAutoTrackTooDense1      = -9086,    // Automation event scripts are too dense.
00174     kFicAutoTrackTooDense2      = -9087,    // Ran out of free events for the automation parser.
00175     kFicNothingAllowedErr        = -9088,    // Missing allowed decks for the hw setup dialog.
00176     kFicHardwareNotFreeErr       = -9089,    // Unable to select a deck because the hardware is
allocated or not available.
00177     kFicUnderrunErr9093          = -9093,    // Under run error from the DSP.
00178     kFicBadVRefNumErr           = -9095,    // Audio file is not on the proper disk SCSI chain.
00179     kFicNoPeripheralSelected     = -9096,    // Deck can not be aquired without a peripheral being
selected.
00180     kFicLaunchMemoryErr         = -9097,    // Unable to launch DAE because of a memory error.
DAE does NOT launch.
00181     kFicGestaltBadSelector       = -9099,    // Gestalt selector not supported.

```

```

00182     kDuplicateWriteFiles           = -9118,    // Writing to the same file multiple times during
processing.
00183     kFicCantGetTempBuffer           = -9121,    // Disk scheduler ran out of temporary buffers.
Playlist is too complex.
00184     kFicPendingRequestsFull         = -9122,    // (Undefined)
00185     kFicRequestHandlesFull          = -9123,    // (Undefined)
00186     kFicAnonymousDrive              = -9124,    // (Win32) Disk scheduler can't use a drive that
doesn't have a drive signature.
00187     kFicComputerNeedsRestart        = -9127,    // DAE state has changed such that the computer needs
to restart
00188     kFicCPUOverload                 = -9128,    // Host processing has exceeded its CPU allocation.
00189     kFicHostInterruptTooLong        = -9129,    // Host processing held off other system interrupts
for too long.
00190     kFicBounceHandlerTooSlow        = -9132,
00191     kFicBounceHandlerTooSlowToConvertWhileBouncing = -9133,
00192     kFicMBoxLostConnection          = -9134,    // MBox was disconnected during playback
00193     kFicMBoxNotConnected             = -9135,    // MBox is not connected
00194     kFicUSBIsynchronousUnderrun     = -9136,    // USB audio streaming underrun
00195     kFicAlreadyAcquired              = -9137,    // tried to change coarse sample rate on already
acquired deck
00196     kFicTDM2BusTopologyErr          = -9138,    // eDsiTDM2BusTopologyErr was returned from DSI.
00197     kFicDirectIODHSAAlreadyOpen     = -9142,    // can't run if a DirectIO client is running
DHS right now
00198     kFicAcquiredButChangedBuffers    = -9143,    // DAE was able to acquire the device but had
to change the disk buffers size to do it.
00199     kFicStreamManagerUnderrun       = -9144,    // received error from StreamManager
00200     kDirectMidiError                 = -9145,    // an error occurred in the DirectMidi
subsystem
00201     kFicResourceForkNotFound        = -9146,    // Could not find the DAE resource fork (i.e.
fnfErr)
00202     kFicInputDelayNotSupported       = -9147,
00203     kFicInsufficientBounceStreams    = -9148,
00204     kFicAutoTotalTooDenseForDSP     = -9155,    // (Undefined)
00205     kBadPlugInSpec                  = -9156,    // Default error returned when there's no component
object attached to a spec.
00206     kFicFarmRequiredErr             = -9157,    // Error returned by objects that require a DSP farm
in the system.
00207     kFicPlugInDidSetCursor           = -9163,    // When returned by FicPlugInEvent, the plug-in
DID change the cursor.
00208     kFicMaxFileCountReached          = -9168,    // Max number of files open has been reached
00209     kFicCantIncreaseAIOLimits        = -9169,    // Can't increase the AIO kernel limits on OSX.
DigiShoeTool is probably not installed correctly.
00210     kFicGreenOverrunWhileVSIOsOn    = -9170,    // A PIO underrun/overrun occurred while
varispeed is on; should probably warn the user this can happen.
00211     kFicBerlinGreenStreamingError    = -9171,
00212     kFicHardwareDeadlineMissedErr    = -9172,
00213     kFicStatePacketUnderrun          = -9173,    // Low-latency engine ran out of state packets
sent from high-latency engine
00214     kFicCannotCompleteRequestError   = -9174,
00215     kFicNILParameterError            = -9175,    // Method called with one or more required
parameters set to NULL
00216     kFicMissingOrInvalidAllowedPlugInsListFile = -9176,    // PT First-specific: could not parse the
"Allowed" plug-ins file
00217     kFicBufferNotLargeEnoughError    = -9177, // Method called with a data buffer that is too small for
the requested data
00218     kFicInitializationFailed          = -9178, // Error caught during FicInit
00219     kFicPostPacketFailed              = -9179, // Error triggered by AAXH_CPlugIn::PostPacket
00220
00221 };
00222
00223 // Weird errors preserved here for backwards compatibility (i.e., older DAE's returned these errors,
so we should also):
00224
00225 enum {
00226     kFicBeyondPlayableRange          = -9735    // Session playback passed the signed 32 bit sample
number limit ( = kFicParseErr - 700).
00227 };
00228
00229
00230 //-----
00231 // Error codes returned from the SADriver/DigiSystem INIT via DAE
00232 //-----
00233
00234 enum {
00235     kFicSADriverErrOffset             = -9200,    // Offset only, should never be returned as a result.
00236     kSADUnsupported                   = -9201,    // Unsupported feature being set from a piece of
hardware.
00237     kSADNoStandardShell               = -9202,    // Unable to load standard shell code resource.
00238     kSADTooManyPeripherals             = -9203,    // Went beyond the max number of peripherals allowed
in the code.
00239     kSADHostTimeoutErr                = -9204,    // Timeout occurred while trying to communicate with
the DSP's host port.
00240     kSADInvalidValue                  = -9205,    // Invalid value being set to a hardware feature.
00241     kSADInvalidObject                 = -9206,    // NULL object found when a valid object is required.
00242
00243     kSADNILClient                     = -9210,    // Trying to operate on a NULL client.
00244     kSADClientRegistered              = -9211,    // Client already registered.

```

```

00245     kSADClientUnregistered      = -9212,    // Trying to remove a client when it's not registered.
00246     kSADNoListener               = -9213,    // No client to respond to a message from another
        client.
00247
00248     kSADCardOwned                = -9220,    // A card is owned by a client.
00249     kSADDSPOwned                 = -9230,    // A DSP is owned by a client.
00250
00251     kSADNILShell                 = -9240,    // Trying to operate on a NULL shell.
00252     kSADShellRegistered          = -9241,    // Shell already registered.
00253     kSADShellUnregistered        = -9242,    // Trying to remove a shell when it's not registered.
00254     kSADShellTooSmall            = -9243,    // (Undefined)
00255     kSADShellTooLarge            = -9244,    // DSP code runs into standard shell or runs out of P
        memory.
00256     kSADStandardShell            = -9245,    // Trying to unregister the standard shell.
00257
00258     kSADNoDriverFile             = -9250,    // Unable to open or create the DigiSetup file.
00259     kSADDriverFileUnused         = -9251,    // Trying to free the DigiSetup file when it hasn't
        been opened.
00260     kSADNILResource              = -9252,    // Resource not found in the DigiSetup file.
00261     kSADBadSize                  = -9253,    // Resource size does not match pointer size
        requested.
00262     kSADBadSlot                  = -9254,    // NuBus slot value is out of range for the system.
00263     kSADBadIndex                 = -9255,    // DSP index is out of range for the system.
00264 };
00265
00266
00267 //-----
00268 // Error codes for Elastic audio
00269 //-----
00270 enum {
00271     kFicElasticGeneralErr        = -9400,    // don't know what else to do
00272     kFicElasticUnsupported       = -9401,    // requested op unsupported
00273     kFicElasticCPUOverload       = -9403,    // Like kFicCPUOverload but for Fela
00274     kFicElasticOutOfMemory       = -9404,    // you're not going to last long...
00275     kFicElasticTrackTooDense     = -9405,    // like kFicAudioTrackTooDense; feeder list too big
00276     kFicElasticInadequateBuffering = -9406,    // reserved buffers for Fela data too small
00277     kFicElasticConnectionErr     = -9408,    // Problem with a plugin connection
00278     kFicElasticDriftBackwardsErr = -9411,    // disconnect between DAE (app?) and plugin data
        consumption rates
00279     kFicElasticDriftForwardsErr  = -9412,    // disconnect between DAE (app?) and plugin data
        consumption rates
00280     kFicElasticPluginLimitsErr   = -9413,    // problem with plugin drift/lookAhead; too much
        requested?
00281     kFicElasticInvalidParameter  = -9415,    // Elastic function was passed a bad parameter
00282     kFicElasticInvalidState      = -9416,    // Elastic track's internal state is in error.
00283     kFicElasticPluginConnected   = -9417,    // Can't change stem format once an elastic plugin is
        already connected to a track
00284     kFicElasticEphemeralAllocErr = -9419,    // ephemeral buffer alloc failure
00285     kFicElasticDiskTooSlowErr    = -9473,    // Like -9073, but caught in a new way (Elastic needs
        disk data sooner)
00286 };
00287
00288 //-----
00289 // Error codes for Clip Gain RT Fades
00290 //-----
00291 enum {
00292     kFicClipGainRTFadesFadeOutofBounds = 9480,
00293 };
00294
00295 //-----
00296 // Error codes for Disk Cache
00297 //-----
00298 enum {
00299     kFicDiskCachePageOverflow     = -9500,    // not enough pages in the cache to fulfill page
        request.
00300     kFicDiskCacheWriteErr        = -9502,    // problem writing to the disk cache.
00301     kFicDiskCacheDiskWriteErr    = -9503,    // problem writing to disk from the cache.
00302     kFicDiskCacheInvalidNull     = -9504,    // invalid NULL variable. NULL and 0 have special
        meaning in the cache.
00303     kFicDiskCacheMissingDataErr  = -9506,    // data that's supposed to be in the cache is not.
00304     kFicDiskCacheGeneralErr      = -9507,    // general error.
00305     kFicDiskCacheDoubleLRUPageErr = -9508,    // duplicate page in the LRU.
00306     kFicDiskCacheDoubleOwnerPageErr = -9509,    // two pages with the same owner.
00307     kFicDiskCachePageLeakErr     = -9510,    // page leak in the allocator.
00308     kFicDiskCacheMappingErr      = -9511,    // corruption in mapping of disk cache objects to the
        page allocator
00309     kFicDiskCacheUnityFileErr    = -9513,    // Unity and ISIS are incompatible with the disk
        cache's temporary buffers
00310     kFicDiskCacheOutOfMemory     = -9514,    // Couldn't allocate the disk cache! 32bits will
        suffocate us all.
00311     kFicNativeDiskCacheOutOfMemory = -9515,    // Couldn't allocate the disk cache on a Native
        system!
00312 };
00313
00314 //-----
00315 // Error codes for FPGA DMA Device (Green and Berlin cards)
00316 //-----

```

```

00317 enum {
00318     kFicFpgaDmaDevicePIOOverflow          = -9600,    // PIO ring buffer overflowed
00319     kFicFpgaDmaDevicePIOUnderflow         = -9601,    // PIO ring buffer underflow
00320     kFicFpgaDmaDevicePIOSyncErr           = -9602,    // PIO sync error
00321     kFicFpgaDmaDevicePIOClockChange       = -9603,    // PIO clock change error
00322     kFicFpgaDmaDevicePIOUnknownErr        = -9604,    // PIO unknown error
00323     kFicFpgaDmaDeviceTDMRcvOverflow       = -9605,    // TDM receive overflow
00324     kFicFpgaDmaDeviceTDMXmtUnderflow      = -9606,    // TDM transmit underflow
00325     kFicFpgaDmaDeviceTDMSyncErr           = -9607,    // TDM sync error
00326     kFicFpgaDmaDeviceTDMCRCErr            = -9608,    // TDM CRC error
00327     kFicFpgaDmaDeviceTDM_NO_Xbar_Txdata_error = -9609,    // TDM NO_Xbar_Txdata_error
00328     kFicFpgaDmaDeviceTDMUnknownErr        = -9610,    // TDM unknown error
00329     kFicFpgaDmaDeviceRegRdTimeoutErr      = -9611,    // RegRdTimeoutErr
00330     kFicFpgaDmaDeviceTemperatureErr       = -9612,    // Temperature error
00331 };
00332
00333 //-----
00334 // Various Widget Error Codes
00335 //-----
00336
00337 enum {
00338     // External Callback Proc Errors -7000..-7024
00339     kSelectorNotSupported                  = -7000,    // This selector ID is unknown currently.
00340     kWidgetNotFound                       = -7001,    // Refnum did not specify a known widget.
00341
00342     // Plug-In Manager Errors -7025..-7049
00343     kPlugInNotInstantiated                = -7026,    // A non-instantiated plug-in was asked to do
00344     something.
00345     kNilComponentObject                   = -7027,    // A component-referencing object was NIL.
00346     kWidgetNotOpen                       = -7028,    // A non-instantiated widget was asked to do
00347     something.
00347     //TIMILEONE ADD
00348     kDspMgrError                         = -7030,    // An error originating in DspMgr returned
00349     kEffectInstantiateError               = -7032,    // Problem occurred attempting to instantiate a
00350     plug-in.
00351     // Plug-In Manager Errors -7050..-7075
00352     kNotEnoughHardware                   = -7050,    // Not enough hardware available to instantiate a
00353     plug-in.
00353     kNotEnoughTDMSlots                   = -7052,    // Not enough TDM slots available to instantiate a
00354     plug-in.
00354     kCantInstantiatePlugIn               = -7054,    // Unable to instantiate a plug-in (generic error).
00355     kCantFindPlugIn                     = -7055,    // Unable to find the specified plug-in.
00356     kNoPlugInsExist                     = -7056,    // No plug-ins at all exist.
00357     kPlugInUnauthorized                  = -7058,    // To catch uncopyprotected plugins
00358     kInvalidHostSignalNet                = -7062,    // The signalNet ptr does not correspond to a
00359     CHostSignalNet instance
00359     // The RTAS/TDM plug-in would be disabled because the corresponding AAX plug-in exists.
00360     //
00361     // The following lower-level errors can also be converted to kPlugInDisabled:
00362     // kAAXH_Result_FailedToRegisterEffectPackageWrongArchitecture
00363     // kAAXH_Result_PluginBuiltAgainstIncompatibleSDKVersion
00364     kPlugInDisabled                     = -7063,
00365     kPlugInNotAllowed                   = -7064,    // The plug-in not allowed to load
00366
00367     // Widget errors (returned by calls to widget functions): -7075..-7099.
00368     kWidgetUnsupportedSampleRate         = -7081,    // Widget cannot instantiate at the current sample
00369     rate
00370     // Connection errors: -7100..-7124
00371     kInputPortInUse                     = -7100,    // Tried to connect to an input that is already
00372     connected.
00372     kOutputPortCannotConnect             = -7101,    // Specified output port has reached its limit of
00373     output connections.
00373     kInvalidConnection                   = -7103,    // Invalid or freed connection reference passed.
00374     kBadConnectionInfo                   = -7104,    // TDM talker & listener data not consistent on
00375     disconnect.
00375     kFreeConnectionErr                   = -7105,    // Could not delete connection info.
00376     kInvalidPortNum                     = -7106,    // Out-of-range or nonexistent port number specified.
00377     kPortIsDisconnected                  = -7107,    // Tried to disconnect a disconnected port.
00378
00379     kBadStemFormat                       = -7110,
00380     kBadInputStemFormat                  = -7111,
00381     kBadOutputStemFormat                 = -7112,
00382     kBadSideChainStemFormat              = -7113,
00383     kBadGenericStemFormat                = -7114,
00384     kBadUnknownStemFormat                = -7115,
00385
00386     kNoFirstRTASDuringPlayback           = -7117,    // can't instantiate the first RTAS plug-in on the fly
00387     (TDM decks)
00387     kNoBridgeConnectionDuringPlayback    = -7118,    // can't create or free a bridge connection during
00388     playback
00389     // Subwidget errs: -7125..-7149
00390     kInstanceIndexRangeErr               = -7126,    // Specified instance index doesn't correspond with an
00391     instance.

```

```

00391     kEmptySubWidgetList           = -7129,    // List isn't NULL, but has no elements.
00392
00393     // Instance errs: -7150..-7174
00394     kNumInstancesWentNegative      = -7150,    // Somehow a count of instances (in widget or DSP)
went < 0.
00395     kCantChangeNumInputs           = -7152,    // Plugin does not have variable number of inputs.
00396     kCantChangeNumOutputs          = -7153,    // Plugin does not have variable number of outputs.
00397     kSetNumInputsOutOfRange         = -7154,    // Number of inputs being set is out of range.
00398     kSetNumOutputsOutOfRange        = -7155,    // Number of outputs being set is out of range.
00399     kChunkRangeErr                 = -7157,    // Handle of plugin settings will not work on a
plugin.
00400
00401     // driver call errs: -7200..-7249
00402     kBadDriverRefNum                = -7200,    // Plugin does not have a valid driver object.
00403     kBadHardwareRefNum              = -7201,    // Plugin does not have a valid pointer to a hardware
object. DSPPtr = NULL.
00404     kBadWidgetRef                   = -7202,    // Widget object is NULL.
00405     kLoggedExceptionInConnMgr       = -7224,    // Logged exception caught in Connection Manager
00406     kUnknownExceptionInConnMgr      = -7225,    // Unknown exception caught in Connection Manager
00407
00408     // Widget control errors: -7300..-7324
00409     kControlIndexRangeErr           = -7300,    // Passed control index was out of range (couldn't
find control).
00410     kNotOurControl                  = -7301,    // Passed in control that didn't belong to widget.
00411     kNullControl                    = -7302,    // Passed in control ref was NULL.
00412     kControlNumStepsErr              = -7303,    // Control provided an invalid number of steps
00413
00414     // Builtin plugin errors: -7350..-7374
00415     kUnsupportedBuiltinPlugin        = -7350,    // Invalid built-in plugin spec.
00416     kAssertErr                      = -7400,
00417
00418     // ASP Processing errors: - 7450..-7499
00419     kFicProcessStuckInLoop           = -7450,    // Plugin is stuck in a loop for an process pass.
00420     kFicOutputBoundsNotInitd         = -7452,    // Plugin needs to set output connections to valid
range within InitOutputBounds.
00421     kFicConnectionBufferOverwrite    = -7453,    // Plugin overwrote the end of the connection buffer.
00422     kFicNoASPBouds                  = -7454,    // Start and end bounds for an ASP process or analysis
were equal.
00423     kFicASPDoneProcessing             = -7456,    // The ASP terminated processing with no errors.
00424     kFicASPErrorWritingToDisk         = -7457,    // ASP encountered error while writing audio data to
disk.
00425     kFicASPOutputFileTooLarge         = -7458,    // ASP tried to write a file larger than the 2^31
bytes in size.
00426     kFicASPOverwriteOnUnity           = -7459,    // ASP tried to write destructively to Unity
00427
00428     // Errors called from Failure Handler routines.
00429     kUnknownErr                      = -7401    // Plugin caught an unknown exception
00430 };
00431
00432 //-----
00433 // Digi Serial Port Errors
00434 //-----
00435
00436 enum {
00437     kFicSerBadParameterPointer        = -7500,
00438     kFicSerBadRoutineSelector          = -7501,
00439     kFicSerPortDoesNotExist            = -7502,
00440     kFicSerPortAlreadyInUse            = -7503,
00441     kFicSerPortNotOpen                 = -7504,
00442     kFicSerBadPortRefereceNumber       = -7505
00443 };
00444
00445 // Play nice with emacs
00446 // Local variables:
00447 // mode:c++
00448 // End:
00449
00450 */
00451
00452
00453 // AAXH.h
00454 /*
00455 enum
00456 {
00457     kAAXH_Result_NoErr = 0,
00458     kAAXH_Result_Error_Base = -14000,           // ePSError_Base_AAXHost
// kAAXH_Result_Error = kAAXH_Result_Error_Base - 0,
00459     kAAXH_Result_Warning = kAAXH_Result_Error_Base - 1,
00460     kAAXH_Result_UnsupportedPlatform = kAAXH_Result_Error_Base - 3,
00461     kAAXH_Result_EffectNotRegistered = kAAXH_Result_Error_Base - 4,
00462     kAAXH_Result_IncompleteInstantiationRequest = kAAXH_Result_Error_Base - 5,
00463     kAAXH_Result_NoShellMgrLoaded = kAAXH_Result_Error_Base - 6,
00464     kAAXH_Result_UnknownExceptionLoadingTIPlugIn = kAAXH_Result_Error_Base - 7,
00465     kAAXH_Result_EffectComponentsMissing = kAAXH_Result_Error_Base - 8,
00466     kAAXH_Result_BadLegacyPlugInIDIndex = kAAXH_Result_Error_Base - 9,
00467     kAAXH_Result_EffectFactoryInitdTooManyTimes = kAAXH_Result_Error_Base - 10,
00468     kAAXH_Result_InstanceNotFoundWhenDeinstantiating = kAAXH_Result_Error_Base - 11,

```

```

00470     kAXXH_Result_FailedToRegisterEffectPackage =    kAXXH_Result_Error_Base - 12,
00471     kAXXH_Result_PluginSignatureNotValid =         kAXXH_Result_Error_Base - 13,
00472     kAXXH_Result_ExceptionDuringInstantiation =     kAXXH_Result_Error_Base - 14,
00473     kAXXH_Result_ShuffleCancelled =                 kAXXH_Result_Error_Base - 15,
00474     kAXXH_Result_NoPacketTargetRegistered =         kAXXH_Result_Error_Base - 16,
00475     kAXXH_Result_ExceptionReconnectingAfterShuffle = kAXXH_Result_Error_Base - 17,
00476     kAXXH_Result_EffectModuleCreationFailed =       kAXXH_Result_Error_Base - 18,
00477     kAXXH_Result_AccessingUninitializedComponent =  kAXXH_Result_Error_Base - 19,
00478     kAXXH_Result_TTComponentInstantiationPostponed = kAXXH_Result_Error_Base - 20,
00479     kAXXH_Result_FailedToRegisterEffectPackageNotAuthorized = kAXXH_Result_Error_Base - 21,
00480     kAXXH_Result_FailedToRegisterEffectPackageWrongArchitecture = kAXXH_Result_Error_Base - 22,
00481     kAXXH_Result_PluginBuiltAgainstIncompatibleSDKVersion = kAXXH_Result_Error_Base - 23,
00482     kAXXH_Result_RequiredPropertyMissing =          kAXXH_Result_Error_Base - 24,
00483     kAXXH_Result_ObjectCopyFailed =                 kAXXH_Result_Error_Base - 25,
00484     kAXXH_Result_CouldNotGetPluginBundleLoc =        kAXXH_Result_Error_Base - 26,
00485     kAXXH_Result_CouldNotFindExecutableInBundle =    kAXXH_Result_Error_Base - 27,
00486     kAXXH_Result_CouldNotGetExecutableLoc =          kAXXH_Result_Error_Base - 28,
00487
00488     kAXXH_Result_InvalidArgumentValue =              kAXXH_Result_Error_Base - 100, // WARNING:
00489     Overlaps with eTISysErrorBase
00489     kAXXH_Result_NameNotFoundInPageTable =           kAXXH_Result_Error_Base - 101 // WARNING:
00489     Overlaps with eTISysErrorNotImpl
00490 };
00491
00492 */
00493
00494
00495 // PlatformSupport_Error.h
00496 /*
00497 enum
00498 {
00499     ePSError_None = 0,
00500     ePSError_Base_DSI = -1000, // DaeStatus.h
00501     ePSError_Base_DirectIO = -6000, // DirectIODefs.h
00502     ePSError_Base_DirectMIDI = -6500, // DirectIODefs.h
00503
00504     ePSError_Base_DAE_Plugins = -7000, // FicErrors.h
00505     ePSError_Base_DAE_Disk = -8000, // FicErrors.h
00506     ePSError_Base_DAE_General = -9000, // FicErrors.h
00507     ePSError_Base_DAE_DCM = -11000, // FicErrors.h
00508     ePSError_General_PLEASESTOPUSINGTHIS = -12000,
00509     ePSError_Generic_PLEASESTOPUSINGTHIS = -12001,
00510     ePSError_OutOfMemory = -12002,
00511     ePSError_OutOfHardwareMemory = -12003,
00512     ePSError_FixedListTooSmall = -12004,
00513     ePSError_FileNotFound = -12005,
00514     ePSError_Timeout = -12006,
00515     ePSError_FileReadError = -12007,
00516     ePSError_InvalidArgs = -12008,
00517
00518     ePSError_DEXBase_Interrupts = -12100,
00519     ePSError_DEXBase_PCI = -12200,
00520     ePSError_DEXBase_Task = -12300,
00521     ePSError_DEXBase_Console = -12400,
00522     ePSError_Base_PalmerEngine = -12500,
00523     ePSError_Base_IP = -12600,
00524     ePSError_Base_DEXLoader = -12700,
00525     ePSError_Base_DEXDebugger = -12800,
00526     ePSError_Base_DEXDLLLoader = -12900,
00527     ePSError_Base_Thread = -13000,
00528     ePSError_Base_Hardware = -13100,
00529     ePSError_Base_TMS = -13400, // TMSErrors.h
00530     ePSError_Base_Harpo = -13500, // DhM_HarpoInterface.h
00531     ePSError_Base_FlashProgram = -13600, // Hampton_HostFPGAProgramming.h
00532     ePSError_Base_Balance = -13700, // DhM_Balance.h
00533     ePSError_Base_CTIDSP = -13800, // DhM_Core_TIDSP.h
00534     ePSError_Base_ONFPGASerial = -13900, // DhM_CONFPGASerialController.h
00535     ePSError_Base_AAXHost = -14000, // AAXH.h
00536     ePSError_Base_TISys = -14100, // TISysError.h
00537     ePSError_Base_DIDL = -14200, // DIDL.h
00538     ePSError_Base_TIDSPMgr = -14300, // TIDSPMgrAllocationReturnCodes.h
00539     ePSError_Base_Berlin = -14400, // DhM_Berlin.h
00540     ePSError_Base_Isoch = -14500, // DhM_IsochEngine.h
00541     ePSError_SupplHW_NotSupported = -14600, // DhM_SupplHW.h
00542
00543     // Add new ranges here...
00544
00545     ePSError_Base_AAXPlugIns = -20000, // AAX_Errors.h
00546
00547     ePSError_Base_DynamicErrors = -30000, // Dynamically Generated error tokens
00548
00549
00550
00551     ePSError_Base_GenericErrorTranslations = -21000, // these errors used to be
00552     ePSError_Generic_PLEASESTOPUSINGTHIS - splitting into unique error codes
00552     // putting this out in space in case anyone's using other numbers on another branch
00553     ePSError_CEthDCMDeviceInterface_CreatePort_UncaughtException

```

```
= -21001,
00554     ePSError_CEthDCMDeviceInterface_DestroyPort_UncaughtException
= -21002,
00555     ePSError_CEEPro1000Imp_InitializeAndAllocateBuffers_NullE1000State
= -21003,
00556     ePSError_CIODeviceOverviewsManager_OvwDataThreadNull
= -21004,
00557     ePSError_CIODeviceOverviewsManager_ThreadAlreadyRunning
= -21005,
00558     ePSError_CPalmerEngineKernelImp_CreateIsochronousStream_PalmerEngineIsCurrentlyShuttingDown
= -21006,
00559     ePSError_CPalmerEngineKernelImp_SetStreamEnabledState_PalmerEngineIsCurrentlyShuttingDown
= -21007,
00560     ePSError_CPalmerEngineImplementation_StartOperating_DidNotFindPartnerInTime
= -21008,
00561     ePSError_CPalmerEngineImplementation_TransmitAsyncMessage_PalmerEngineIsCurrentlyShuttingDown
= -21009,
00562     ePSError_CPalmerEngineImplementation_TransmitAsyncMessageAndWaitForReply_PalmerEngineIsCurrentlyShuttingDown
= -21010,
00563     ePSError_CPalmerEngineImplementation_TransmitAsyncMessageAndWaitForReply_PalmerEngineIsShuttingDownAfterReply
= -21011,
00564     ePSError_CPalmerEngineImplementation_TransmitGeneralAsyncPacket_PalmerEngineIsShuttingDown
= -21012,
00565     ePSError_CEthernetDeviceSimpleImp_InitializeAndAllocateBuffers_FailedToGetBufferInfo
= -21013,
00566     ePSError_CPEInterface_Imp_GetFeatureSetList_FailedWinGetResourceOfModuleByName
= -21014,
00567     ePSError_CPEInterface_Imp_GetFourPartVersion_FailedWinGetVersionOfModuleByName
= -21015,
00568     ePSError_CHamptonHostDEXLifeLine_Common_TransmitMessageAndGetReply_TransmitAndWaitForReplyFailed
= -21016,
00569     ePSError_CHamptonHostDEXLifeLine_Common_TransmitMessageAndGetReply_ConnectionClosedOrNotEstablished
= -21017,
00570     ePSError_CHamptonHostDEXLifeLine_Common_TransmitMessageAndGetReply_GotUnexpectedReply
= -21018,
00571     ePSError_PerformLoadNotSupportedOnMac
= -21019,
00572     ePSError_PerformLoad_FailedGetUnusedUDPPort
= -21020,
00573     ePSError_PerformLoad_FailedCreateLocalUDPEndPoint
= -21021,
00574     ePSError_PerformLoad_FailedCreateRemoteUDPEndPoint
= -21022,
00575     ePSError_PerformLoad_FailedToGetPacketFromEndpoint
= -21023,
00576     ePSError_PerformLoad_FirstPacketContainsUnexpectedData
= -21024,
00577     ePSError_PerformLoad_SecondPacketContainsUnexpectedData
= -21025,
00578     ePSError_PerformLoad_FailedToGetCorrectPacketFromEndpoint
= -21026,
00579     ePSError_HamptonDEXLoader_LoadOverUDP_UpdateImageBootInterfaceHeaderFailed
= -21027,
00580     ePSError_HamptonDEXLoader_ResetOverUDP_FailedGetUnusedUDPPort
= -21028,
00581     ePSError_HamptonDEXLoader_ResetOverUDP_FailedCreateLocalUDPEndPoint
= -21029,
00582     ePSError_CTask_Imp_SetSchedulingParameters_FailedThreadSpecificDataInit
= -21030,
00583     ePSError_CTask_Imp_SetSchedulingParameters_FailedToSetFirstThreadPriority
= -21031,
00584     ePSError_CTask_Imp_SetSchedulingParameters_FailedToSetSecondThreadPriority
= -21032,
00585     ePSError_CTask_Imp_SetSchedulingParameters_FailedToVerifyNewPolicy
= -21033,
00586     ePSError_CTask_Imp_SetSchedulingParameters_FailedToSetTimeshareToFalse
= -21034,
00587     ePSError_CTask_Imp_SetSchedulingParameters_FailedToGetThreadPolicy
= -21035,
00588     ePSError_CTask_Imp_SetSchedulingParameters_FailedToSetThirdThreadPriority
= -21036,
00589     ePSError_CTask_Imp_SetSchedulingParameters_FailedToGetThreadPolicyAgain
= -21037,
00590     ePSError_CModule_Hardware_Imp_GetHardwareMemoryAvailable_WinError
= -21038,
00591     ePSError_CModule_Hardware_Imp_SetHardwareMemoryRequired_WinError
= -21039,
00592     ePSError_Win_CModule_Hardware_Imp_MapAndGetDALDevices_MapIOCTLFailed
= -21040,
00593     ePSError_CModule_Hardware_Imp_ThreadMethod_CreateDALHandleFailed
= -21041,
00594     ePSError_CSynchPrim_Semaphore_Imp_CSynchPrim_Semaphore_Imp_CreateSemaphoreFailed
= -21042,
00595     ePSError_CSynchPrim_Event_Imp_CSynchPrim_Event_Imp_CreateEventFailed
```



```

    = -21043,
00596     ePSError_CTask_Imp_SetSchedulingParameters_gSetInfoThreadProcNotSet
    = -21044,
00597     ePSError_CTask_Imp_SetSchedulingParameters_SetThreadPriorityFailed
    = -21045,
00598     ePSError_CTask_Imp_SetProcessorAffinityMask_SetThreadAffinityMaskFailed
    = -21046,
00599     ePSError_PSThreadTable_VerifyTableEntryExists_NotFound
    = -21047,
00600     ePSError_PSM_SimpleThread_ThreadMethod_RunThrewException
    = -21048,
00601     ePSError_Hampton_DEXImage_MakeROM_BadFilename
    = -21049,
00602     ePSError_MakeDllIntoHex_BadFilename
    = -21050,
00603     ePSError_MakeDllIntoHex_BadPayloadObject
    = -21051,
00604     ePSError_MakeDllIntoHex_FailedCreatePEInterface
    = -21052,
00605     ePSError_MakeDllIntoHex_FailedResolvedAllSymbols
    = -21053,
00606     ePSError_MakeDllIntoHexWithStdCLib_BadFilename
    = -21054,
00607     ePSError_MakeDllIntoHexWithStdCLib_NULLDEXImages
    = -21055,
00608     ePSError_CDEXWin32Kernel_ExceptionsModule_Initialize_FailedToCreateTLSContext
    = -21056,
00609     ePSError_CDEXIP_ARP_Imp_GetMACForGivenIP_IPAddressMaskBad
    = -21057,
00610     ePSError_CDEXIP_ARP_Imp_GetMACForGivenIP_IPAddressInvalid
    = -21058,
00611     ePSError_DEXIntegrityCheck_VerifySection_FailureCheckingSectionCookies
    = -21059,
00612     ePSError_DEXIntegrityCheck_VerifySection_FailureCheckingSectionBufferCookie
    = -21060,
00613     ePSError_DEXIntegrityCheck_VerifyTextSection_FailedChecksum
    = -21061,
00614     ePSError_Mac_CModule_Hardware_Imp_MapAndGetDALDevices_MapIOCTLFailed
    = -21062,
00615     ePSError_CModule_Hardware_Imp_ThreadMethod_mach_port_allocate_failed
    = -21063,
00616     ePSError_DEXTool_main_ExceptionThrown
    = -21064,
00617     ePSError_Hampton_DEXImage_MakeHexIntoBin_HEXFileNameVersion_StandardExceptionThrown
    = -21065,
00618     ePSError_Hampton_DEXImage_MakeHexIntoBin_HEXFileNameVersion_UnknownExceptionThrown
    = -21066,
00619     ePSError_Hampton_DEXImage_MakeHexIntoBin_HEXDataVersion_StandardExceptionThrown
    = -21067,
00620     ePSError_Hampton_DEXImage_MakeHexIntoBin_HEXDataVersion_UnknownExceptionThrown
    = -21068
00621 };
00622 */
00623
00624 // TISysError.h
00625 /*
00630 enum
00631 {
00632     eTISysErrorSuccess                = 0,                ///< success code
00633     eTISysErrorBase                   = ePSError_Base_TISys,    ///< -14100 see
PlatformSupport_Error.h
00634     eTISysErrorNotImpl                = eTISysErrorBase - 1,    ///< not
implemented
00635     eTISysErrorMemory                 = eTISysErrorBase - 2,    ///< out of memory
00636     eTISysErrorParam                  = eTISysErrorBase - 3,    ///< invalid
parameter
00637     eTISysErrorNull                   = eTISysErrorBase - 4,    ///< NULL value
00638     eTISysErrorCommunication          = eTISysErrorBase - 5,    ///< Communication
problem with Shell
00639     eTISysErrorIllegalAccess          = eTISysErrorBase - 6,
00640     eTISysErrorDirectAccessOfFifoBlocksUnsupported = eTISysErrorBase - 7,
00641     eTISysErrorPortIdOutOfBounds      = eTISysErrorBase - 8,
00642     eTISysErrorPortTypeDoesNotSupportDirectAccess = eTISysErrorBase - 9,
00643     eTISysErrorPIFOfull                = eTISysErrorBase - 10,   ///< FIFO doesn't
have capacity
00644     eTISysErrorRPCTimeOutOnDSP        = eTISysErrorBase - 11,
00645     eTISysErrorShellMgrChip_SegsDontMatchAddr = eTISysErrorBase - 12,
00646     eTISysErrorOnChipRPCNotRegistered = eTISysErrorBase - 13,
00647     eTISysErrorUnexpectedBufferLength  = eTISysErrorBase - 14,
00648     eTISysErrorUnexpectedEntryPointName = eTISysErrorBase - 15,
00649     eTISysErrorPortIDTooLargeForContextBlock = eTISysErrorBase - 16,
00650     eTISysErrorMixerDelayNotSupportedForPlugIns = eTISysErrorBase - 17,
00651     eTISysErrorShellFailedToStartUp    = eTISysErrorBase - 18,
00652     eTISysErrorUnexpectedCondition     = eTISysErrorBase - 19,
00653     eTISysErrorShellNotRunningWhenExpected = eTISysErrorBase - 20,
00654     eTISysErrorFailedToCreateNewPIInstance = eTISysErrorBase - 21,
00655     eTISysErrorUnknownPIInstance       = eTISysErrorBase - 22,

```



```

00656     eTISysErrorTooManyInstancesForSingleBufferProcessing = eTISysErrorBase - 23,
00657     eTISysErrorNoDSPs = eTISysErrorBase - 24,
00658     eTISysBadDSPID = eTISysErrorBase - 25,
00659     eTISysBadPContextWriteBlockSize = eTISysErrorBase - 26,
00660     eTISysInstanceInitFailed = eTISysErrorBase - 28,
00661     eTISysSameModuleLoadedTwiceOnSameChip = eTISysErrorBase - 29,
00662     eTISysCouldNotOpenPlugInModule = eTISysErrorBase - 30,
00663     eTISysPlugInModuleMissingDependencies = eTISysErrorBase - 31,
00664     eTISysPlugInModuleLoadableSegmentCountMismatch = eTISysErrorBase - 32,
00665     eTISysPlugInModuleLoadFailure = eTISysErrorBase - 33,
00666     eTISysOutOfOnChipDebuggingSpace = eTISysErrorBase - 34,
00667     eTISysMissingAlgEntryPoint = eTISysErrorBase - 35,
00668     eTISysInvalidRunningStatus = eTISysErrorBase - 36,
00669     eTISysExceptionRunningInstantiation = eTISysErrorBase - 37,
00670     eTISysTIShellBinaryNotFound = eTISysErrorBase - 38,
00671     eTISysTimeoutWaitingForTIShell = eTISysErrorBase - 39,
00672     eTISysSwapScriptTimeout = eTISysErrorBase - 40,
00673     eTISysTIDSPModuleNotFound = eTISysErrorBase - 41,
00674     eTISysTIDSPReadError = eTISysErrorBase - 42,
00675
00676 };
00677
00678 */
00679
00681 #endif // AAX_ERRORS_H

```

15.133 AAX_Exception.h File Reference

```

#include "AAX_Assert.h"
#include "AAX_StringUtilities.h"
#include "AAX.h"
#include <exception>
#include <string>
#include <set>

```

15.133.1 Description

AAX SDK exception classes and utilities

Classes

- class [AAX::Exception::Any](#)
- class [AAX::Exception::ResultError](#)
- class [AAX_CheckedResult](#)
- class [AAX_AggregateResult](#)

Namespaces

- namespace [AAX](#)
- namespace [AAX::Exception](#)
AAX exception classes

Macros

- `#define AAX_SWALLOW(...)`
Executes *X* in a try/catch block that catches [AAX_CheckedResult](#) exceptions.
- `#define AAX_SWALLOW_MULT(...)`
Executes *X* in a try/catch block that catches [AAX_CheckedResult](#) exceptions.
- `#define AAX_CAPTURE(X, ...)`
Executes *Y* in a try/catch block that catches [AAX::Exception::ResultError](#) exceptions and captures the result.
- `#define AAX_CAPTURE_MULT(X, ...)`
Executes *Y* in a try/catch block that catches [AAX::Exception::ResultError](#) exceptions and captures the result.

Functions

- `std::string AAX::AsString` (const char *inStr)
- `const std::string & AAX::AsString` (const std::string &inStr)
- `const std::string & AAX::AsString` (const Exception::Any &inStr)

15.133.2 Macro Definition Documentation

15.133.2.1 AAX_SWALLOW

```
#define AAX_SWALLOW(
    ... )
```

Value:

```
try { if(true) { ( __VA_ARGS__ ); } } \
catch (const AAX_CheckedResult::Exception& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
    AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (swallowed)", \
        __FILE__, __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
} do {} while (false)
```

Executes *X* in a try/catch block that catches [AAX_CheckedResult](#) exceptions.

Catches exceptions thrown from [AAX_CheckedResult](#) only - other exceptions require an explicit catch.

```
AAX_CheckedResult cr;
cr = NecessaryFunc1();
AAX_SWALLOW(cr = FailableFunc());
cr = NecessaryFunc2();
```

15.133.2.2 AAX_SWALLOW_MULT

```
#define AAX_SWALLOW_MULT(
    ... )
```

Value:

```
try { if(true) { __VA_ARGS__ } } \
catch (const AAX_CheckedResult::Exception& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
    AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (swallowed)", __FILE__, \
        __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
} do {} while (false)
```

Executes *X* in a try/catch block that catches [AAX_CheckedResult](#) exceptions.

Version of [AAX_SWALLOW](#) for multi-line input.

Catches exceptions thrown from [AAX_CheckedResult](#) only - other exceptions require an explicit catch.

```
AAX_CheckedResult cr;
cr = NecessaryFunc();
AAX_SWALLOW_MULT(
    cr = FailableFunc1();
    cr = FailableFunc2(); // may not execute
    cr = FailableFunc3(); // may not execute
);
cr = NecessaryFunc2();
```

15.133.2.3 AAX_CAPTURE

```
#define AAX_CAPTURE(
    X,
    ... )
```

Value:

```
try { if(true) { ( __VA_ARGS__ ); } } \
catch (const AAX::Exception::ResultError& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (captured)", __FILE__,
    __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
(X) = AAX_PREPROCESSOR_CONCAT(ex,__LINE__).Result(); \
} do {} while (false)
```

Executes `Y` in a try/catch block that catches `AAX::Exception::ResultError` exceptions and captures the result.

Catches exceptions thrown from `AAX_CheckedResult` and other `AAX::Exception::ResultError` exceptions.

`X` must be an `AAX_Result`

```
AAX_Result result = AAX_SUCCESS;
AAX_CAPTURE(result, ResultErrorThrowingFunc());
// result now holds the error code thrown by ThrowingFunc()

AAX_CheckedResult cr;
AAX_CAPTURE(result, cr = FailableFunc());
```

15.133.2.4 AAX_CAPTURE_MULT

```
#define AAX_CAPTURE_MULT(
    X,
    ... )
```

Value:

```
try { if(true) { __VA_ARGS__ } } \
catch (const AAX_CheckedResult::Exception& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (captured)", __FILE__,
    __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
(X) = AAX_PREPROCESSOR_CONCAT(ex,__LINE__).Result(); \
} do {} while (false)
```

Executes `Y` in a try/catch block that catches `AAX::Exception::ResultError` exceptions and captures the result.

Version of `AAX_CAPTURE` for multi-line input.

Catches exceptions thrown from `AAX_CheckedResult` and other `AAX::Exception::ResultError` exceptions.

`X` must be an `AAX_Result` or an implicitly convertible type

```
AAX_Result result = AAX_SUCCESS;
AAX_CAPTURE_MULT(result,
    MaybeThrowingFunc1();
    MaybeThrowingFunc2();

    // can use AAX_CheckedResult within AAX_CAPTURE_MULT
    AAX_CheckedResult cr;
    cr = FailableFunc1();
    cr = FailableFunc2();
    cr = FailableFunc3();
);

// result now holds the value of the last thrown error
return result;
```

15.134 AAX_Exception.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004   * All rights reserved.
00005   *
00006   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007   * not disclose to any third party. Use of the information contained in this
00008   * document is subject to an Avid SDK license.
00009   */
00010
00016  /*=====*/
00017
00018
00019 #ifndef AAXLibrary_AAX_Exception_h
00020 #define AAXLibrary_AAX_Exception_h
00021
00022 #include "AAX_Assert.h"
00023 #include "AAX_StringUtilities.h"
00024 #include "AAX.h"
00025
00026 #include <exception>
00027 #include <string>
00028 #include <set>
00029
00030
00032 #if 0
00033 #pragma mark -
00034 #pragma mark AAX::Exception
00035 #endif
00037 namespace AAX
00038 {
00039     namespace Exception {
00040         class Any;
00041     }
00042
00043     inline std::string AsString(const char* inStr);
00044     inline const std::string& AsString(const std::string& inStr);
00045     inline const std::string& AsString(const Exception::Any& inStr);
00046
00047
00048     namespace Exception
00049     {
00050
00051         class Any
00052         {
00053         public:
00054             virtual ~Any() {}
00055
00056             template <class C>
00057             explicit Any(const C& inWhat)
00058             : mDesc(AAX::AsString(inWhat))
00059             , mFunction()
00060             , mLine()
00061             , mWhat(AAX::Exception::Any::CreateWhat(mDesc, mFunction, mLine))
00062             {
00063             }
00064
00065             template <class C1, class C2, class C3>
00066             explicit Any(const C1& inWhat, const C2& inFunction, const C3& inLine)
00067             : mDesc(AAX::AsString(inWhat))
00068             , mFunction(AAX::AsString(inFunction))
00069             , mLine(AAX::AsString(inLine))
00070             , mWhat(AAX::Exception::Any::CreateWhat(mDesc, mFunction, mLine))
00071             {
00072             }
00073
00074             // assignment operator
00075             Any& operator=(const Any& inOther)
00076             {
00077                 mDesc = inOther.mDesc;
00078                 mFunction = inOther.mFunction;
00079                 mLine = inOther.mLine;
00080                 mWhat = inOther.mWhat;
00081                 return *this;
00082             }
00083
00084             AAX_DEFAULT_MOVE_CTOR(Any);
00085             AAX_DEFAULT_MOVE_OPER(Any);
00086
00087             public: // AAX::Exception::Any
00088
00089 #ifndef AAX_CPP11_SUPPORT

```

```

00116         // implicit conversion to std::string (mostly for AsString())
00117         operator const std::string&(void) const { return mWhat; }
00118 #endif
00119
00120         const std::string& What() const { return mWhat; }
00121         const std::string& Desc() const { return mDesc; }
00122         const std::string& Function() const { return mFunction; }
00123         const std::string& Line() const { return mLine; }
00124
00125     private:
00126         static std::string CreateWhat(const std::string& inDesc, const std::string& inFunc, const
std::string& inLine)
00127         {
00128             std::string whatStr(inDesc);
00129             if (false == inFunc.empty()) { whatStr += (" func:" + inFunc); }
00130             if (false == inLine.empty()) { whatStr += (" line:" + inLine); }
00131             return whatStr;
00132         }
00133
00134     private:
00135         std::string mDesc;
00136         std::string mFunction;
00137         std::string mLine;
00138         std::string mWhat;
00139     };
00140
00141     class ResultError : public Any
00142     {
00143     public:
00144         explicit ResultError(AAX_Result inWhatResult)
00145             : Any(ResultError::FormatResult(inWhatResult))
00146             , mResult(inWhatResult)
00147             {
00148             }
00149
00150         template <class C>
00151         explicit ResultError(AAX_Result inWhatResult, const C& inFunction)
00152             : Any(ResultError::FormatResult(inWhatResult), inFunction, (const char*)NULL)
00153             , mResult(inWhatResult)
00154             {
00155             }
00156
00157         template <class C1, class C2>
00158         explicit ResultError(AAX_Result inWhatResult, const C1& inFunction, const C2& inLine)
00159             : Any(ResultError::FormatResult(inWhatResult), inFunction, inLine)
00160             , mResult(inWhatResult)
00161             {
00162             }
00163
00164         static std::string FormatResult(AAX_Result inResult)
00165         {
00166             return std::string(AAX::AsStringResult(inResult) + " (" +
AAX::AsStringInt32((int32_t)inResult) + ")");
00167         }
00168
00169         AAX_Result Result() const { return mResult; }
00170
00171     private:
00172         AAX_Result mResult;
00173     };
00174
00175     std::string AsString(const char* inStr)
00176     {
00177         return inStr ? std::string(inStr) : std::string();
00178     }
00179
00180     const std::string& AsString(const std::string& inStr)
00181     {
00182         return inStr;
00183     }
00184
00185     const std::string& AsString(const Exception::Any& inStr)
00186     {
00187         return inStr.What();
00188     }
00189
00190 #if 0
00191 #pragma mark -
00192 #endif
00193
00194     class AAX_CheckedResult
00195     {
00196     public:
00197         typedef AAX::Exception::ResultError Exception;

```

```

00317
00318     /* non-virtual destructor */ ~AAX_CheckedResult() {}
00319
00321     AAX_CheckedResult()
00322     : mCurResult(AAX_SUCCESS)
00323     , mLastError(AAX_SUCCESS)
00324     , mAcceptedResults()
00325     {
00326         Initialize();
00327     }
00328
00331     AAX_CheckedResult(AAX_Result inResult)
00332     : mCurResult(inResult)
00333     , mLastError(AAX_SUCCESS)
00334     , mAcceptedResults()
00335     {
00336         Initialize();
00337         Check();
00338     }
00339
00345     void AddAcceptedResult(AAX_Result inResult)
00346     {
00347         mAcceptedResults.insert(inResult);
00348     }
00349
00350     void ResetAcceptedResults()
00351     {
00352         mAcceptedResults.clear();
00353         mAcceptedResults.insert(AAX_SUCCESS);
00354     }
00355
00357     AAX_CheckedResult& operator=(AAX_Result inResult)
00358     {
00359         mCurResult = inResult;
00360         Check();
00361         return *this;
00362     }
00363
00366     AAX_CheckedResult& operator|=(AAX_Result inResult)
00367     {
00368         return this->operator=(inResult);
00369     }
00370
00372     operator AAX_Result() const
00373     {
00374         return mCurResult;
00375     }
00376
00379     void Clear()
00380     {
00381         mCurResult = AAX_SUCCESS;
00382         mLastError = AAX_SUCCESS;
00383     }
00384
00387     AAX_Result GetLastError() const
00388     {
00389         return mLastError;
00390     }
00391
00392 private:
00393     void Initialize()
00394     {
00395         ResetAcceptedResults();
00396     }
00397
00398     void Check()
00399     {
00400         const AAX_Result err = mCurResult;
00401         if (0 == mAcceptedResults.count(err))
00402         {
00403             AAX_CheckedResult::Exception ex(err);
00404
00405             // error state is now captured in ex
00406             mCurResult = AAX_SUCCESS;
00407             mLastError = err;
00408
00409             AAX_TRACE_RELEASE(kAAX_TracePriority_Normal, "AAX_CheckedResult - throwing %s",
00410 ex.What().c_str());
00411             AAX_STACKTRACE(kAAX_TracePriority_Lowest, ""); // stacktrace is only printed for debug
00412             // plug-in builds
00413             throw ex;
00414         }
00415     }
00416 private:
00417     AAX_Result mCurResult;
00418     AAX_Result mLastError;

```

```

00418     std::set<AAX_Result> mAcceptedResults;
00419 };
00420
00421
00422 #if 0
00423 #pragma mark -
00424 #pragma mark AAX exception macros
00425 #endif
00426
00427 #define AAX_SWALLOW(...) \
00428     try { if(true) { ( __VA_ARGS__ ); } } \
00429     catch (const AAX_CheckedResult::Exception& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
00430         AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (swallowed)", \
00431             __FILE__, __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
00432     } do {} while (false)
00433
00434 #define AAX_SWALLOW_MULT(...) \
00435     try { if(true) { __VA_ARGS__ } } \
00436     catch (const AAX_CheckedResult::Exception& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
00437         AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (swallowed)", \
00438             __FILE__, __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
00439     } do {} while (false)
00440
00441 #define AAX_CAPTURE(X, ...) \
00442     try { if(true) { ( __VA_ARGS__ ); } } \
00443     catch (const AAX::Exception::ResultError& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
00444         AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (captured)", \
00445             __FILE__, __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
00446         (X) = AAX_PREPROCESSOR_CONCAT(ex,__LINE__).Result(); \
00447     } do {} while (false)
00448
00449 #define AAX_CAPTURE_MULT(X, ...) \
00450     try { if(true) { __VA_ARGS__ } } \
00451     catch (const AAX_CheckedResult::Exception& AAX_PREPROCESSOR_CONCAT(ex,__LINE__)) { \
00452         AAX_TRACE_RELEASE(kAAX_Trace_Priority_High, "%s line %d (%s) exception caught: %s (captured)", \
00453             __FILE__, __LINE__, __FUNCTION__, AAX_PREPROCESSOR_CONCAT(ex,__LINE__).What().c_str()); \
00454         (X) = AAX_PREPROCESSOR_CONCAT(ex,__LINE__).Result(); \
00455     } do {} while (false)
00456
00457 #if 0
00458 #pragma mark -
00459 #endif
00460
00461 class AAX_AggregateResult
00462 {
00463 public:
00464     AAX_AggregateResult() = default;
00465
00466     ~AAX_AggregateResult()
00467     {
00468         if (0 == mNumSucceeded && 0 < mNumFailed) {
00469             try {
00470                 // do normal logging
00471                 this->Check();
00472             }
00473             catch(...)
00474             {
00475                 // can't throw from a destructor
00476             }
00477         }
00478     }
00479
00480     AAX_AggregateResult& operator=(AAX_Result inResult)
00481     {
00482         if (AAX_SUCCESS == inResult)
00483         {
00484             ++mNumSucceeded;
00485         }
00486         else
00487         {
00488             mLastFailure = inResult;
00489             ++mNumFailed;
00490         }
00491         return *this;
00492     }
00493
00494     operator AAX_Result()
00495     {
00496         AAX_Result const err = this->LastFailure();
00497         this->Clear();
00498         return err;
00499     }
00500
00501     void Check() const { AAX_CheckedResult tempErr(mLastFailure); }
00502     void Clear() {

```

```

00633         mLastFailure = AAX_SUCCESS;
00634         mNumFailed = 0;
00635         mNumSucceeded = 0;
00636     }
00637
00638     AAX_Result LastFailure() const { return mLastFailure; }
00639     int NumFailed() const { return mNumFailed; }
00640     int NumSucceeded() const { return mNumSucceeded; }
00641     int NumAttempted() const { return mNumFailed+mNumSucceeded; }
00642
00643 private:
00644     AAX_Result mLastFailure{AAX_SUCCESS};
00645     int mNumFailed{0};
00646     int mNumSucceeded{0};
00647 };
00648
00649 #endif

```

15.135 AAX_Exports.cpp File Reference

```

#include "AAX_Init.h"
#include "AAX.h"
#include "acfunknown.h"
#include "acfresult.h"

```

Macros

- #define [AAX_EXPORT](#) extern "C" __declspec(dllexport) ACFRESULT __stdcall

Functions

- [AAX_EXPORT ACFRegisterPlugin](#) (IACFUnknown *pUnkHostVoid, IACFPluginDefinition **ppPluginDefinitionVoid)
The main plug-in registration method.
- [AAX_EXPORT ACFRegisterComponent](#) (IACFUnknown *pUnkHost, acfUInt32 index, IACFComponentDefinition **ppComponentDefinition)
Registers a specific component in the DLL.
- [AAX_EXPORT ACFGetClassFactory](#) (IACFUnknown *pUnkHost, const acfCLSID &clsid, const [acfIID](#) &iid, void **ppOut)
Gets the factory for a given class ID.
- [AAX_EXPORT ACFCanUnloadNow](#) (IACFUnknown *pUnkHost)
Determines whether or not the host may unload the DLL.
- [AAX_EXPORT ACFStartup](#) (IACFUnknown *pUnkHost)
DLL initialization routine.
- [AAX_EXPORT ACFShutdown](#) (IACFUnknown *pUnkHost)
DLL shutdown routine.
- [AAX_EXPORT ACFGetSDKVersion](#) (acfUInt64 *oSDKVersion)
Returns the DLL's SDK version.

15.135.1 Macro Definition Documentation

15.135.1.1 AAX_EXPORT

```
#define AAX_EXPORT extern "C" __declspec(dllexport) ACFRESULT __stdcall
```

15.135.2 Function Documentation

15.135.2.1 ACFRegisterPlugin()

```
ACFAPI ACFRegisterPlugin (
    IACFUnknown * pUnkHost,
    IACFPluginDefinition ** ppPluginDefinition )
```

The main plug-in registration method.

References [AAXRegisterPlugin\(\)](#).

Here is the call graph for this function:



15.135.2.2 ACFRegisterComponent()

```
ACFAPI ACFRegisterComponent (
    IACFUnknown * pUnkHost,
    acfUInt32 index,
    IACFComponentDefinition ** ppComponentDefinition )
```

Registers a specific component in the DLL.

References [AAXRegisterComponent\(\)](#).

Here is the call graph for this function:



15.135.2.3 ACFGetClassFactory()

```
ACFAPI ACFGetClassFactory (
    IACFUnknown * pUnkHost,
    const acfCLSID & clsid,
    const acfIID & iid,
    void ** ppOut )
```

Gets the factory for a given class ID.

References [AAXGetClassFactory\(\)](#).

Here is the call graph for this function:



15.135.2.4 ACFCanUnloadNow()

```
ACFAPI ACFCanUnloadNow (
    IACFUnknown * pUnkHost )
```

Determines whether or not the host may unload the DLL.

References [AAXCanUnloadNow\(\)](#).

Here is the call graph for this function:



15.135.2.5 ACFStartup()

```
ACFAPI ACFStartup (
    IACFUnknown * pUnkHost )
```

DLL initialization routine.

References [AAXStartup\(\)](#).

Here is the call graph for this function:



15.135.2.6 ACFSShutdown()

```
ACFAPI ACFSShutdown (
    IACFUnknown * pUnkHost )
```

DLL shutdown routine.

References [AAXShutdown\(\)](#).

Here is the call graph for this function:



15.135.2.7 ACFGetSDKVersion()

```
ACF_API ACFGetSDKVersion (
    acfUInt64 * oSDKVersion )
```

Returns the DLL's SDK version.

References [AAXGetSDKVersion\(\)](#).

Here is the call graph for this function:



15.136 AAX_GUITypes.h File Reference

```
#include "AAX.h"
#include <AAX_ALIGN_FILE_BEGIN>
#include <AAX_ALIGN_FILE_HOST>
#include <AAX_ALIGN_FILE_END>
#include <AAX_ALIGN_FILE_RESET>
```

15.136.1 Description

Constants and other definitions used by AAX plug-in GUIs.

Classes

- struct [AAX_Point](#)
Data structure representing a two-dimensional coordinate point.
- struct [AAX_Rect](#)
Data structure representing a rectangle in a two-dimensional coordinate plane.

Typedefs

- typedef struct [AAX_Point](#) [AAX_Point](#)
Data structure representing a two-dimensional coordinate point.
- typedef struct [AAX_Rect](#) [AAX_Rect](#)
Data structure representing a rectangle in a two-dimensional coordinate plane.
- typedef enum [AAX_EViewContainer_Type](#) [AAX_EViewContainer_Type](#)
Type of [view container](#).

Enumerations

- enum [AAX_EViewContainer_Type](#) {
[AAX_eViewContainer_Type_NULL](#) = 0 ,
[AAX_eViewContainer_Type_NSView](#) = 1 ,
[AAX_eViewContainer_Type_UIView](#) = 2 ,
[AAX_eViewContainer_Type_HWND](#) = 3 }

Type of *view container*.

Functions

- bool [operator==](#) (const [AAX_Point](#) &p1, const [AAX_Point](#) &p2)
- bool [operator!=](#) (const [AAX_Point](#) &p1, const [AAX_Point](#) &p2)
- bool [operator<](#) (const [AAX_Point](#) &p1, const [AAX_Point](#) &p2)
- bool [operator<=](#) (const [AAX_Point](#) &p1, const [AAX_Point](#) &p2)
- bool [operator>](#) (const [AAX_Point](#) &p1, const [AAX_Point](#) &p2)
- bool [operator>=](#) (const [AAX_Point](#) &p1, const [AAX_Point](#) &p2)
- bool [operator==](#) (const [AAX_Rect](#) &r1, const [AAX_Rect](#) &r2)
- bool [operator!=](#) (const [AAX_Rect](#) &r1, const [AAX_Rect](#) &r2)
- [AAX_ENUM_SIZE_CHECK](#) ([AAX_EViewContainer_Type](#))

15.136.2 Typedef Documentation

15.136.2.1 AAX_Point

```
typedef struct AAX\_Point AAX\_Point
```

Data structure representing a two-dimensional coordinate point.

Comparison operators give preference to `vert`

15.136.2.2 AAX_Rect

```
typedef struct AAX\_Rect AAX\_Rect
```

Data structure representing a rectangle in a two-dimensional coordinate plane.

15.136.2.3 AAX_EViewContainer_Type

```
typedef enum AAX\_EViewContainer\_Type AAX\_EViewContainer\_Type
```

Type of *view container*.

See also

[AAX_IViewContainer::GetType\(\)](#)

15.136.3 Enumeration Type Documentation

15.136.3.1 AAX_EViewContainer_Type

enum [AAX_EViewContainer_Type](#)

Type of [view container](#).

See also

[AAX_IViewContainer::GetType\(\)](#)

Enumerator

AAX_eViewContainer_Type_NULL	
AAX_eViewContainer_Type_NSView	
AAX_eViewContainer_Type_UIView	
AAX_eViewContainer_Type_HWND	

15.136.4 Function Documentation

15.136.4.1 operator==() [1/2]

```
bool operator== (
    const AAX\_Point & p1,
    const AAX\_Point & p2 ) [inline]
```

References [AAX_Point::horz](#), and [AAX_Point::vert](#).

15.136.4.2 operator!=() [1/2]

```
bool operator!= (
    const AAX\_Point & p1,
    const AAX\_Point & p2 ) [inline]
```

15.136.4.3 operator<()

```
bool operator< (
    const AAX_Point & p1,
    const AAX_Point & p2 ) [inline]
```

References [AAX_Point::horz](#), and [AAX_Point::vert](#).

15.136.4.4 operator<=()

```
bool operator<= (
    const AAX_Point & p1,
    const AAX_Point & p2 ) [inline]
```

References [AAX_Point::horz](#), and [AAX_Point::vert](#).

15.136.4.5 operator>()

```
bool operator> (
    const AAX_Point & p1,
    const AAX_Point & p2 ) [inline]
```

15.136.4.6 operator>=()

```
bool operator>= (
    const AAX_Point & p1,
    const AAX_Point & p2 ) [inline]
```

15.136.4.7 operator==() [2/2]

```
bool operator== (
    const AAX_Rect & r1,
    const AAX_Rect & r2 ) [inline]
```

References [AAX_Rect::height](#), [AAX_Rect::left](#), [AAX_Rect::top](#), and [AAX_Rect::width](#).

15.136.4.8 operator"!=()" [2/2]

```
bool operator!= (
    const AAX_Rect & r1,
    const AAX_Rect & r2 ) [inline]
```

15.136.4.9 AAX_ENUM_SIZE_CHECK()

```
AAX_ENUM_SIZE_CHECK (
    AAX_EViewContainer_Type )
```

15.137 AAX_GUITypes.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2015, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020
00021
00023 #ifndef AAX_GUIYPES_H
00024 #define AAX_GUIYPES_H
00026
00027 #ifndef _TMS320C6X
00028
00029 #include "AAX.h"
00030
00031 #include AAX_ALIGN_FILE_BEGIN
00032 #include AAX_ALIGN_FILE_HOST
00033 #include AAX_ALIGN_FILE_END
00038     typedef struct AAX_Point
00039     {
00040         AAX_Point (
00041             float v,
00042             float h) :
00043             vert(v),
00044             horz(h)
00045         {}
00046
00047         AAX_Point (
00048             void) :
00049             vert(0.0f),
00050             horz(0.0f)
00051         {}
00052
00053         float vert;
00054         float horz;
00055     } AAX_Point;
00056 #include AAX_ALIGN_FILE_BEGIN
00057 #include AAX_ALIGN_FILE_RESET
00058 #include AAX_ALIGN_FILE_END
00059
00060 inline bool operator==(const AAX_Point& p1, const AAX_Point& p2)
00061 {
00062     return ((p1.vert == p2.vert) && (p1.horz == p2.horz));
00063 }
00064
00065 inline bool operator!=(const AAX_Point& p1, const AAX_Point& p2)
00066 {
00067     return !(p1 == p2);
00068 }
```



```

00069
00070 inline bool operator<(const AAX_Point& p1, const AAX_Point& p2)
00071 {
00072     return (p1.vert == p2.vert) ? (p1.horz < p2.horz) : (p1.vert < p2.vert);
00073 }
00074
00075 inline bool operator<=(const AAX_Point& p1, const AAX_Point& p2)
00076 {
00077     return (p1.vert == p2.vert) ? (p1.horz <= p2.horz) : (p1.vert < p2.vert);
00078 }
00079
00080 inline bool operator>(const AAX_Point& p1, const AAX_Point& p2)
00081 {
00082     return !(p1 <= p2);
00083 }
00084
00085 inline bool operator>=(const AAX_Point& p1, const AAX_Point& p2)
00086 {
00087     return !(p1 < p2);
00088 }
00089
00090 #include AAX_ALIGN_FILE_BEGIN
00091 #include AAX_ALIGN_FILE_HOST
00092 #include AAX_ALIGN_FILE_END
00093     typedef struct AAX_Rect
00094     {
00095         AAX_Rect (
00096             float t,
00097             float l,
00098             float w,
00099             float h) :
00100             top(t),
00101             left(l),
00102             width(w),
00103             height(h)
00104         {}
00105
00106         AAX_Rect (
00107             void) :
00108             top(0.0f),
00109             left(0.0f),
00110             width(0.0f),
00111             height(0.0f)
00112         {}
00113
00114         float top;
00115         float left;
00116         float width;
00117         float height;
00118     } AAX_Rect;
00119 #include AAX_ALIGN_FILE_BEGIN
00120 #include AAX_ALIGN_FILE_RESET
00121 #include AAX_ALIGN_FILE_END
00122
00123 inline bool operator==(const AAX_Rect& r1, const AAX_Rect& r2)
00124 {
00125     return ((r1.top == r2.top) && (r1.left == r2.left) && (r1.width == r2.width) && (r1.height ==
00126         r2.height));
00127 }
00128
00129 inline bool operator!=(const AAX_Rect& r1, const AAX_Rect& r2)
00130 {
00131     return !(r1 == r2);
00132 }
00133
00134 typedef enum AAX_EViewContainer_Type
00135 {
00136     AAX_eViewContainer_Type_NULL = 0,
00137     AAX_eViewContainer_Type_NSView = 1,
00138     AAX_eViewContainer_Type_UIView = 2,
00139     AAX_eViewContainer_Type_HWND = 3
00140 } AAX_EViewContainer_Type;
00141 AAX_ENUM_SIZE_CHECK( AAX_EViewContainer_Type );
00142
00143 #endif // _TMS320C6X
00144
00145 #endif // AAX_GUITYPES_H

```

15.138 AAX_IACFAutomationDelegate.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

15.138.1 Description

Versioned interface allowing an AAX plug-in to interact with the host's automation system.

Classes

- class [AAX_IACFAutomationDelegate](#)

Versioned interface allowing an AAX plug-in to interact with the host's automation system.

15.139 AAX_IACFAutomationDelegate.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IACFAUTOMATIONDELEGATE_H
00023 #define AAX_IACFAUTOMATIONDELEGATE_H
00024
00025 #include "AAX.h"
00026
00027 #ifdef __clang__
00028 #pragma clang diagnostic push
00029 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00030 #endif
00031
00032 #include "acfunknown.h"
00033
00040 class AAX_IACFAutomationDelegate : public IACFUnknown
00041 {
00042 public:
00043
00046     virtual AAX_Result      RegisterParameter ( AAX_CParamID iParameterID ) = 0;
00047
00050     virtual AAX_Result      UnregisterParameter ( AAX_CParamID iParameterID ) = 0;
00051
00054     virtual AAX_Result      PostSetValueRequest ( AAX_CParamID iParameterID, double normalizedValue )
00055     const = 0;
00058     virtual AAX_Result      PostCurrentValue( AAX_CParamID iParameterID, double normalizedValue ) const
00059     = 0;
00062     virtual AAX_Result      PostTouchRequest( AAX_CParamID iParameterID ) = 0;
00063
00066     virtual AAX_Result      PostReleaseRequest( AAX_CParamID iParameterID ) = 0;
00067
00070     virtual AAX_Result      GetTouchState ( AAX_CParamID iParameterID, AAX_CBoolean * oTouched )= 0;
00071 };
00072
00073 #ifdef __clang__
00074 #pragma clang diagnostic pop
00075 #endif
00076
00077 #endif // AAX_IACFAUTOMATIONDELEGATE_H
```

15.140 AAX_IACFCollection.h File Reference

```
#include "acfbaseapi.h"
```

15.140.1 Description

Versioned interface to represent a plug-in binary's static description.

Classes

- class [AAX_IACFCollection](#)

Versioned interface to represent a plug-in binary's static description.

15.141 AAX_IACFCollection.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IACFCOLLECTION_H
00023 #define AAX_IACFCOLLECTION_H
00024
00025 #ifdef __clang__
00026 #pragma clang diagnostic push
00027 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00028 #endif
00029
00030 #include "acfbaseapi.h"
00031
00032 class AAX_IEffectDescriptor;
00033
00036 class AAX_IACFCollection : public IACFPluginDefinition
00037 {
00038 public:
00039
00040     virtual AAX_Result          AddEffect ( const char * inEffectID, IACFUnknown *
inEffectDescriptor ) = 0;
00041     virtual AAX_Result          SetManufacturerName( const char* inPackageName ) = 0;
00042     virtual AAX_Result          AddPackageName( const char *inPackageName ) = 0;
00043     virtual AAX_Result          SetPackageVersion( uint32_t inVersion ) = 0;
00044     virtual AAX_Result          SetProperties ( IACFUnknown * inProperties ) = 0;
00045 };
00046
00047 #ifdef __clang__
00048 #pragma clang diagnostic pop
00049 #endif
00050
00051 #endif
```

15.142 AAX_IACFComponentDescriptor.h File Reference

```
#include "AAX.h"
#include "AAX_Callbacks.h"
#include "AAX_IDma.h"
#include "acfunknown.h"
```

15.142.1 Description

Versioned description interface for an AAX plug-in algorithm callback.

Classes

- class [AAX_IACFComponentDescriptor](#)
Versioned description interface for an AAX plug-in algorithm callback.
- class [AAX_IACFComponentDescriptor_V2](#)
Versioned description interface for an AAX plug-in algorithm callback.
- class [AAX_IACFComponentDescriptor_V3](#)
Versioned description interface for an AAX plug-in algorithm callback.

15.143 AAX_IACFComponentDescriptor.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef _AAX_IACFCOMPONENTDESCRIPTOR_H_
00023 #define _AAX_IACFCOMPONENTDESCRIPTOR_H_
00024
00025 #include "AAX.h"
00026 #include "AAX_Callbacks.h"
00027 #include "AAX_IDma.h"
00028
00029 #ifdef __clang__
00030 #pragma clang diagnostic push
00031 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00032 #endif
00033
00034 #include "acfunknown.h"
00035
00036
00039 class AAX_IACFComponentDescriptor : public IACFUnknown
00040 {
00041 public:
00042     virtual AAX_Result Clear () = 0;
00043     virtual AAX_Result AddReservedField ( AAX_CFieldIndex inFieldIndex, uint32_t inFieldType ) = 0;
00044     virtual AAX_Result AddAudioIn ( AAX_CFieldIndex inFieldIndex ) = 0;
00045     virtual AAX_Result AddAudioOut ( AAX_CFieldIndex inFieldIndex ) = 0;
00046     virtual AAX_Result AddAudioBufferLength ( AAX_CFieldIndex inFieldIndex ) = 0;
00047     virtual AAX_Result AddSampleRate ( AAX_CFieldIndex inFieldIndex ) = 0;
00048     virtual AAX_Result AddClock ( AAX_CFieldIndex inFieldIndex ) = 0;
```

```

00049     virtual AAX_Result  AddSideChainIn ( AAX_CFieldIndex inFieldIndex ) = 0;
00050     virtual AAX_Result  AddDataInPort ( AAX_CFieldIndex inFieldIndex, uint32_t inPacketSize,
AAX_EDataInPortType inPortType) = 0;
00051     virtual AAX_Result  AddAuxOutputStem ( AAX_CFieldIndex inFieldIndex, int32_t inStemFormat, const
char inNameUTF8[]) = 0;
00052     virtual AAX_Result  AddPrivateData ( AAX_CFieldIndex inFieldIndex, int32_t inDataSize, uint32_t
inOptions = AAX_ePrivateDataOptions_DefaultOptions ) = 0;
00053     virtual AAX_Result  AddDmaInstance ( AAX_CFieldIndex inFieldIndex, AAX_IDma::EMode inDmaMode ) =
0;
00054     virtual AAX_Result  AddMIDINode ( AAX_CFieldIndex inFieldIndex, AAX_EMIDINodeType inNodeType,
const char inNodeName[], uint32_t channelMask ) = 0;
00055
00056     virtual AAX_Result  AddProcessProc_Native (
00057         AAX_CProcessProc inProcessProc,
00058         IACFUnknown * inProperties,
00059         AAX_CInstanceInitProc inInstanceInitProc,
00060         AAX_CBackgroundProc inBackgroundProc,
00061         AAX_CSelector * outProcID) = 0;
00062     virtual AAX_Result  AddProcessProc_TI (
00063         const char inDLLFileNameUTF8 [],
00064         const char inProcessProcSymbol [],
00065         IACFUnknown * inProperties,
00066         const char inInstanceInitProcSymbol [],
00067         const char inBackgroundProcSymbol [],
00068         AAX_CSelector * outProcID) = 0;
00069
00070     virtual AAX_Result  AddMeters ( AAX_CFieldIndex inFieldIndex, const AAX_CTypeID* inMeterIDs,
const uint32_t inMeterCount ) = 0;
00071 };
00072
00075 class AAX_IACFComponentDescriptor_V2 : public AAX_IACFComponentDescriptor
00076 {
00077 public:
00078     virtual AAX_Result  AddTemporaryData( AAX_CFieldIndex inFieldIndex, uint32_t inDataElementSize) =
0;
00079 };
00080
00083 class AAX_IACFComponentDescriptor_V3 : public AAX_IACFComponentDescriptor_V2
00084 {
00085 public:
00086     virtual AAX_Result  AddProcessProc (
00087         IACFUnknown * inProperties,
00088         AAX_CSelector* outProcIDs,
00089         int32_t inProcIDsSize) = 0;
00090 };
00091
00092 #ifdef __clang__
00093 #pragma clang diagnostic pop
00094 #endif
00095
00096 #endif // #ifndef _AAX_IACFCOMPONENTDESCRIPTOR_H_

```

15.144 AAX_IACFController.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

15.144.1 Description

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.

Classes

- class [AAX_IACFController](#)
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.
- class [AAX_IACFController_V2](#)

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.

- class [AAX_IACFController_V3](#)

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.

15.145 AAX_IACFController.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00020 /*=====*/
00021
00022
00023 #ifndef _AAX_IACFCONTROLLER_H_
00024 #define _AAX_IACFCONTROLLER_H_
00025
00026 #include "AAX.h"
00027
00028 #ifdef __clang__
00029 #pragma clang diagnostic push
00030 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00031 #endif
00032
00033 #include "acfunknown.h"
00034
00035 // Forward declarations
00036 class AAX_IPageTable;
00037 class AAX_IString;
00038
00042 class AAX_IACFController : public IACFUnknown
00043 {
00044 public:
00045
00046     // Host information getters
00047     virtual
00048     AAX_Result
00049     GetEffectID (
00050         AAX_IString *    outEffectID) const = 0;
00051
00052     virtual
00053     AAX_Result
00054     GetSampleRate (
00055         AAX_CSampleRate *outSampleRate ) const = 0;
00056
00057     virtual
00058     AAX_Result
00059     GetInputStemFormat (
00060         AAX_EStemFormat *outStemFormat ) const = 0;
00061
00062     virtual
00063     AAX_Result
00064     GetOutputStemFormat (
00065         AAX_EStemFormat *outStemFormat) const = 0;
00066
00067     virtual
00068     AAX_Result
00069     GetSignalLatency(
00070         int32_t* outSamples) const = 0;
00071
00072     virtual
00073     AAX_Result
00074     GetCycleCount(
00075         AAX_EProperty inWhichCycleCount,
00076         AAX_CPropertyValue* outNumCycles) const = 0;
00077
00078     virtual
00079     AAX_Result
00080     GetTODLocation (
```

```

00088         AAX_CTimeOfDay* outTODLocation ) const = 0;
00089
00090         //Host Information Setters (Dynamic info)
00091     virtual
00092     AAX_Result
00093     SetSignalLatency(
00094         int32_t inNumSamples) = 0;
00095
00096
00097     virtual
00098     AAX_Result
00099     SetCycleCount(
00100         AAX_EProperty* inWhichCycleCounts,
00101         AAX_CPropertyValue* iValues,
00102         int32_t numValues) = 0;
00103
00104
00105         // Posting functions
00106     virtual
00107     AAX_Result
00108     PostPacket (
00109         AAX_CFieldIndex inFieldIndex,
00110         const void * inPayloadP,
00111         uint32_t inPayloadSize) = 0;
00112
00113
00114         //Metering functions
00115     virtual
00116     AAX_Result
00117     GetCurrentMeterValue (
00118         AAX_CTypeID inMeterID,
00119         float * outMeterValue ) const = 0;
00120
00121
00122     virtual
00123     AAX_Result
00124     GetMeterPeakValue (
00125         AAX_CTypeID inMeterID,
00126         float * outMeterPeakValue ) const = 0;
00127
00128
00129     virtual
00130     AAX_Result
00131     ClearMeterPeakValue (
00132         AAX_CTypeID inMeterID ) const = 0;
00133
00134
00135     virtual
00136     AAX_Result
00137     GetMeterClipped (
00138         AAX_CTypeID inMeterID,
00139         AAX_CBoolean * outClipped ) const = 0;
00140
00141
00142     virtual
00143     AAX_Result
00144     ClearMeterClipped (
00145         AAX_CTypeID inMeterID ) const = 0;
00146
00147
00148     virtual
00149     AAX_Result
00150     GetMeterCount (
00151         uint32_t * outMeterCount ) const = 0;
00152
00153
00154         // MIDI methods
00155     virtual
00156     AAX_Result
00157     GetNextMIDIPacket (
00158         AAX_CFieldIndex* outPort,
00159         AAX_CMidiPacket* outPacket ) = 0;
00160
00161 };
00162
00163
00164 class AAX_IACFController_V2 : public AAX_IACFController
00165 {
00166 public:
00167     // Notification method
00168     virtual
00169     AAX_Result
00170     SendNotification (
00171         AAX_CTypeID inNotificationType,
00172         const void* inNotificationData,
00173         uint32_t inNotificationDataSize) = 0;
00174
00175
00176     virtual
00177     AAX_Result
00178     GetHybridSignalLatency(
00179         int32_t* outSamples) const = 0;
00180
00181
00182     virtual
00183     AAX_Result
00184     GetCurrentAutomationTimestamp(
00185         AAX_CTransportCounter* outTimestamp) const = 0;
00186
00187
00188
00189

```

```

00191     virtual
00192     AAX_Result
00193     GetHostName(
00194         AAX_IString* outHostNameString) const = 0;
00195 };
00196
00199 class AAX_IACFController_V3 : public AAX_IACFController_V2
00200 {
00201 public:
00202     virtual
00203     AAX_Result
00204     GetPlugInTargetPlatform(
00205         AAX_CTargetPlatform* outTargetPlatform) const = 0;
00206
00207     virtual
00208     AAX_Result
00209     GetIsAudioSuite(AAX_CBoolean* outIsAudioSuite) const = 0;
00210 };
00211
00212 #ifdef __clang__
00213 #pragma clang diagnostic pop
00214 #endif
00215 #endif // #ifndef _AAX_IACFCONTROLLER_H_

```

15.146 AAX_IACFDataBuffer.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFDataBuffer](#)
Versioned interface for reference counted data buffers.

Macros

- #define [AAX_IACFDataBuffer_H](#)

15.146.1 Macro Definition Documentation

15.146.1.1 AAX_IACFDataBuffer_H

```
#define AAX_IACFDataBuffer_H
```


15.147 AAX_IACFDataBuffer.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019
00020 #ifndef AAX_IACFDataBuffer_H
00021 #define AAX_IACFDataBuffer_H
00022
00023 #include "AAX.h"
00024
00025 #ifdef __clang__
00026 #pragma clang diagnostic push
00027 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00028 #endif
00029
00030 #include "acfunknown.h"
00031
00040 class AAX_IACFDataBuffer : public IACFUnknown
00041 {
00042 public:
00049     virtual AAX_Result Type(AAX_CTypeID * oType) const = 0;
00053     virtual AAX_Result Size(int32_t * oSize) const = 0;
00057     virtual AAX_Result Data(void const ** oBuffer) const = 0;
00058 };
00059
00060 #ifdef __clang__
00061 #pragma clang diagnostic pop
00062 #endif
00063
00064 #endif

```

15.148 AAX_IACFDescriptionHost.h File Reference

```

#include "AAX.h"
#include "acfbbaseapi.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFDescriptionHost](#)

15.149 AAX_IACFDescriptionHost.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.

```

```

00009  */
00010
00011 #ifndef AAXLibrary_AAX_IACFDescriptionHost_h
00012 #define AAXLibrary_AAX_IACFDescriptionHost_h
00013
00014
00015 #include "AAX.h"
00016
00017 class AAX_IACFFeatureInfo;
00018
00019 #ifdef __clang__
00020 #pragma clang diagnostic push
00021 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00022 #endif
00023
00024 #include "acfbaseapi.h"
00025 #include "acfunknown.h"
00026
00029 class AAX_IACFDescriptionHost : public IACFUnknown
00030 {
00031 public:
00032     // NOTE: Documentation is not copied directly from AAX_IDescriptionHost due to an adaptation of
00033     // parameter types (IACFUnknown to AAX_IFeatureInfo)
00040     virtual AAX_Result AcquireFeatureProperties(const AAX_Feature_UID& inFeatureID, IACFUnknown**
00041         outFeatureProperties) = 0;
00042
00043 #ifdef __clang__
00044 #pragma clang diagnostic pop
00045 #endif
00046
00047 #endif

```

15.150 AAX_IACFEffectorDescriptor.h File Reference

```

#include "AAX.h"
#include "AAX_Callbacks.h"
#include "acfunknown.h"

```

15.150.1 Description

Versioned interface for an [AAX_IEffectorDescriptor](#).

Classes

- class [AAX_IACFEffectorDescriptor](#)
Versioned interface for an [AAX_IEffectorDescriptor](#).
- class [AAX_IACFEffectorDescriptor_V2](#)
Versioned interface for an [AAX_IEffectorDescriptor](#).

15.151 AAX_IACFEffectorDescriptor.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this

```

```

00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019  /*=====*/
00020
00021
00022 #ifndef AAX_IACFEFFECTDESCRIPTOR_H
00023 #define AAX_IACFEFFECTDESCRIPTOR_H
00024
00025 #include "AAX.h"
00026 #include "AAX_Callbacks.h"
00027
00028 #ifdef __clang__
00029 #pragma clang diagnostic push
00030 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00031 #endif
00032
00033 #include "acfunknown.h"
00034
00035
00038 class AAX_IACFEffEffectDescriptor : public IACFUnknown
00039 {
00040 public:
00041
00042     virtual AAX_Result          AddComponent ( IACFUnknown* inComponentDescriptor )
00043     = 0;
00044     virtual AAX_Result          AddName ( const char *inPlugInName ) = 0;
00045     virtual AAX_Result          AddCategory ( uint32_t inCategory ) = 0;
00046     virtual AAX_Result          AddCategoryBypassParameter ( uint32_t inCategory,
00047     AAX_CParamID inParamID ) = 0;
00048     virtual AAX_Result          AddProcPtr ( void * inProcPtr, AAX_CProcPtrID
00049     inProcID ) = 0;
00050     virtual AAX_Result          SetProperties ( IACFUnknown * inProperties ) = 0;
00051     virtual AAX_Result          AddResourceInfo ( AAX_EResourceType inResourceType,
00052     const char * inInfo ) = 0;
00053     virtual AAX_Result          AddMeterDescription( AAX_CTypeID inMeterID, const
00054     char * inMeterName, IACFUnknown * inProperties ) = 0;
00055 };
00056
00057 class AAX_IACFEffEffectDescriptor_V2 : public AAX_IACFEffEffectDescriptor
00058 {
00059 public:
00060     virtual AAX_Result          AddControlMIDINode ( AAX_CTypeID inNodeID,
00061     AAX_EMIDINodeType inNodeType, const char inNodeName[], uint32_t inChannelMask ) = 0;
00062 };
00063
00064 #ifdef __clang__
00065 #pragma clang diagnostic pop
00066 #endif
00067 #endif // AAX_IACFEFFECTDESCRIPTOR_H

```

15.152 AAX_IACFEffEffectDirectData.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

15.152.1 Description

The direct data access interface that gets exposed to the host application.

Classes

- class [AAX_IACFEffEffectDirectData](#)
Optional interface for direct access to a plug-in's alg memory.
- class [AAX_IACFEffEffectDirectData_V2](#)

15.153 AAX_IACFEEffectDirectData.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012
00019  /*=====*/
00020
00021
00022  #ifndef AAX_IACFEFFECTDIRECTDATA_H
00023  #define AAX_IACFEFFECTDIRECTDATA_H
00024
00025  #include "AAX.h"
00026
00027  #ifdef __clang__
00028  #pragma clang diagnostic push
00029  #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00030  #endif
00031
00032  #include "acfunknown.h"
00033
00034
00044  class AAX_IACFEEffectDirectData : public IACFUnknown
00045  {
00046  public:
00047
00059      virtual AAX_Result      Initialize ( IACFUnknown * iController ) = 0;
00066      virtual AAX_Result      Uninitialize () = 0;
00068
00069
00101      virtual AAX_Result      TimerWakeup (
00102          IACFUnknown *      iDataAccessInterface ) = 0;
00104  };
00105
00106
00107  class AAX_IACFEEffectDirectData_V2 : public AAX_IACFEEffectDirectData{
00108  public:
00137      virtual      AAX_Result      NotificationReceived( /* AAX_ENotificationEvent */ AAX_CTypeID
        inNotificationType, const void * inNotificationData, uint32_t      inNotificationDataSize) = 0;
00139
00140  };
00141
00142  #ifdef __clang__
00143  #pragma clang diagnostic pop
00144  #endif
00145
00146  #endif //AAX_IACFEFFECTDIRECTDATA_H

```

15.154 AAX_IACFEEffectGUI.h File Reference

```

#include "AAX.h"
#include "AAX_GUITypes.h"
#include "AAX_IString.h"
#include "acfunknown.h"

```

15.154.1 Description

The GUI interface that gets exposed to the host application.

Classes

- class [AAX_IACFEEffectGUI](#)

The interface for a AAX Plug-in's GUI (graphical user interface).

15.155 AAX_IACFEEffectGUI.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IACFEFFECTGUI_H
00023 #define AAX_IACFEFFECTGUI_H
00024
00025 #include "AAX.h"
00026 #include "AAX_GUITypes.h"
00027 #include "AAX_IString.h"
00028
00029 #ifdef __clang__
00030 #pragma clang diagnostic push
00031 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00032 #endif
00033
00034 #include "acfunknown.h"
00035
00036
00082 class AAX_IACFEEffectGUI : public IACFUnknown
00083 {
00084 public:
00085
00097     virtual AAX_Result      Initialize ( IACFUnknown * iController ) = 0;
00104     virtual AAX_Result      Uninitialize () = 0;
00106
00135     virtual AAX_Result      NotificationReceived( /* AAX_ENotificationEvent */ AAX_CTypeID
inNotificationType, const void * inNotificationData, uint32_t inNotificationDataSize) = 0;
00137
00147     virtual AAX_Result      SetViewContainer ( IACFUnknown * iViewContainer ) = 0;
00156     virtual AAX_Result      GetViewSize ( AAX_Point * oViewSize ) const = 0;
00158
00165     virtual AAX_Result      Draw ( AAX_Rect * iDrawRect ) = 0;
00180     virtual AAX_Result      TimerWakeup () = 0;
00194     virtual AAX_Result      ParameterUpdated( AAX_CParamID inParamID) = 0;
00196
00197
00214     virtual AAX_Result      GetCustomLabel ( AAX_EPlugInStrings iSelector, AAX_IString * oString )
const = 0;
00232     virtual AAX_Result      SetControlHighlightInfo ( AAX_CParamID iParameterID, AAX_CBoolean
iIsHighlighted, AAX_EHighlightColor iColor ) = 0;
00234
00235 };
00236
00237 #ifdef __clang__
00238 #pragma clang diagnostic pop
00239 #endif
00240
00241 #endif //AAX_IACFEFFECTGUI_H
```

15.156 AAX_IACFEEffectParameters.h File Reference

```
#include "AAX.h"
#include "acfunknown.h"
```

15.156.1 Description

The data model interface that is exposed to the host application.

Classes

- class [AAX_IACFEffEffectParameters](#)
The interface for an AAX Plug-in's data model.
- struct [AAX_SHybridRenderInfo](#)
Hybrid render processing context.
- class [AAX_IACFEffEffectParameters_V2](#)
Supplemental interface for an AAX Plug-in's data model.
- class [AAX_IACFEffEffectParameters_V3](#)
Supplemental interface for an AAX Plug-in's data model.
- class [AAX_IACFEffEffectParameters_V4](#)
Supplemental interface for an AAX Plug-in's data model.

15.157 AAX_IACFEffEffectParameters.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IACFEFFECTPARAMETERS_H
00023 #define AAX_IACFEFFECTPARAMETERS_H
00024
00025 #include "AAX.h"
00026
00027 class AAX_IString;
00028 class AAX_IParameter;
00029
00030
00031 #ifdef __clang__
00032 #pragma clang diagnostic push
00033 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00034 #endif
00035
00036 #include "acfunknown.h"
00037
00053 class AAX_IACFEffEffectParameters : public IACFUnknown
00054 {
00055 public:
00056
00068     virtual AAX_Result Initialize(IACFUnknown * iController) = 0;
00075     virtual AAX_Result Uninitialize () = 0;
00077
00078
00107     virtual AAX_Result NotificationReceived( /* AAX_ENotificationEvent */ AAX_CTypeID
inNotificationType, const void * inNotificationData, uint32_t inNotificationDataSize) = 0;
00109
00110
00129     virtual AAX_Result GetNumberOfParameters ( int32_t * oNumControls ) const = 0;
00138     virtual AAX_Result GetMasterBypassParameter ( AAX_IString * oIDString ) const = 0;
00148     virtual AAX_Result GetParameterIsAutomatable ( AAX_CParamID iParameterID, AAX_CBoolean
* oAutomatable ) const = 0;
00161     virtual AAX_Result GetParameterNumberOfSteps ( AAX_CParamID iParameterID, int32_t *
oNumSteps ) const = 0;

```

```

00172     virtual AAX_Result           GetParameterName ( AAX_CParamID iParameterID, AAX_IString * oName )
const = 0;
00194     virtual AAX_Result           GetParameterNameOfLength ( AAX_CParamID iParameterID, AAX_IString *
oName, int32_t iNameLength ) const = 0;
00204     virtual AAX_Result           GetParameterDefaultNormalizedValue ( AAX_CParamID iParameterID,
double * oValue ) const = 0;
00216     virtual AAX_Result           SetParameterDefaultNormalizedValue ( AAX_CParamID iParameterID,
double iValue ) = 0;
00228     virtual AAX_Result           GetParameterType ( AAX_CParamID iParameterID, AAX_EParameterType *
oParameterType ) const = 0;
00263     virtual AAX_Result           GetParameterOrientation ( AAX_CParamID iParameterID,
AAX_EParameterOrientation * oParameterOrientation ) const = 0;
00279     virtual AAX_Result           GetParameter ( AAX_CParamID iParameterID, AAX_IParameter **
oParameter ) = 0;
00292     virtual AAX_Result           GetParameterIndex ( AAX_CParamID iParameterID, int32_t *
oControlIndex ) const = 0;
00305     virtual AAX_Result           GetParameterIDFromIndex ( int32_t iControlIndex, AAX_IString *
oParameterIDString ) const = 0;
00321     virtual AAX_Result           GetParameterValueInfo ( AAX_CParamID iParameterID, int32_t
iSelector, int32_t* oValue) const = 0;
00323
00324
00325
00351     virtual AAX_Result           GetParameterValueFromString ( AAX_CParamID iParameterID, double *
oValue, const AAX_IString & iValueString ) const = 0;
00371     virtual AAX_Result           GetParameterStringFromValue ( AAX_CParamID iParameterID, double
iValue, AAX_IString * oValueString, int32_t iMaxLength ) const = 0;
00386     virtual AAX_Result           GetParameterValueString ( AAX_CParamID iParameterID, AAX_IString*
oValueString, int32_t iMaxLength ) const = 0;
00396     virtual AAX_Result           GetParameterNormalizedValue ( AAX_CParamID iParameterID, double *
oValuePtr ) const = 0;
00411     virtual AAX_Result           SetParameterNormalizedValue ( AAX_CParamID iParameterID, double
iValue ) = 0;
00441     virtual AAX_Result           SetParameterNormalizedRelative( AAX_CParamID iParameterID, double
iValue ) = 0;
00443
00444
00445
00471     virtual AAX_Result           TouchParameter ( AAX_CParamID iParameterID ) = 0;
00485     virtual AAX_Result           ReleaseParameter ( AAX_CParamID iParameterID ) = 0;
00499     virtual AAX_Result           UpdateParameterTouch ( AAX_CParamID iParameterID, AAX_CBoolean
iTouchState ) = 0;
00501
00502
00536     virtual AAX_Result           UpdateParameterNormalizedValue ( AAX_CParamID iParameterID, double
iValue, AAX_EUpdateSource iSource ) = 0;
00559     virtual AAX_Result           UpdateParameterNormalizedRelative ( AAX_CParamID iParameterID,
double iValue ) = 0;
00577     virtual AAX_Result           GenerateCoefficients() = 0;
00579
00580
00607     virtual AAX_Result           ResetFieldData (AAX_CFieldIndex inFieldIndex, void * oData, uint32_t
inDataSize) const = 0;
00609
00610
00641     virtual AAX_Result           GetNumberOfChunks ( int32_t * oNumChunks ) const = 0;
00651     virtual AAX_Result           GetChunkIDFromIndex ( int32_t iIndex, AAX_CTypeID * oChunkID )
const = 0;
00672     virtual AAX_Result           GetChunkSize ( AAX_CTypeID iChunkID, uint32_t * oSize ) const = 0;
00696     virtual AAX_Result           GetChunk ( AAX_CTypeID iChunkID, AAX_SPlugInChunk * oChunk ) const
= 0;
00710     virtual AAX_Result           SetChunk ( AAX_CTypeID iChunkID, const AAX_SPlugInChunk * iChunk ) =
0;
00729     virtual AAX_Result           CompareActiveChunk ( const AAX_SPlugInChunk * iChunkP, AAX_CBoolean
* oIsEqual ) const = 0;
00747     virtual AAX_Result           GetNumberOfChanges ( int32_t * oNumChanges ) const = 0;
00749
00766     virtual AAX_Result           TimerWakeup( ) = 0;
00768
00813     virtual AAX_Result           GetCurveData( /* AAX_ECurveType */ AAX_CTypeID iCurveType, const
float * iValues, uint32_t iNumValues, float * oValues ) const = 0;
00815
00837     virtual AAX_Result           GetCustomData( AAX_CTypeID iDataBlockID, uint32_t inDataSize, void*
oData, uint32_t* oDataWritten) const = 0;
00838
00848     virtual AAX_Result           SetCustomData( AAX_CTypeID iDataBlockID, uint32_t inDataSize, const
void* iData ) = 0;
00850
00864     virtual AAX_Result           DoMIDITransfers() = 0;
00866 };
00867
00874 struct AAX_SHybridRenderInfo
00875 {
00876     float**           mAudioInputs;
00877     int32_t*          mNumAudioInputs;
00878     float**           mAudioOutputs;
00879     int32_t*          mNumAudioOutputs;

```

```

00880     int32_t*                mNumSamples;
00881     AAX_CTimestamp*         mClock;
00882 };
00883
00884 class AAX_IACFEffEffectParameters_V2 : public AAX_IACFEffEffectParameters
00885 {
00886 public:
00887     virtual AAX_Result      RenderAudio_Hybrid(AAX_SHybridRenderInfo* ioRenderInfo) = 0;
00888     virtual AAX_Result      UpdateMIDINodes ( AAX_CFieldIndex inFieldIndex, AAX_CMidiPacket&
00889 iPacket ) = 0;
00890     virtual AAX_Result      UpdateControlMIDINodes ( AAX_CTypeID nodeID, AAX_CMidiPacket&
00891 iPacket ) = 0;
00892 };
00893
00894 class AAX_IACFEffEffectParameters_V3 : public AAX_IACFEffEffectParameters_V2
00895 {
00896 public:
00897     virtual AAX_Result      GetCurveDataMeterIds ( /* AAX_ECType */ AAX_CTypeID iCurveType,
00898 uint32_t *oXMeterId, uint32_t *oYMeterId) const = 0;
00899     virtual AAX_Result      GetCurveDataDisplayRange( /* AAX_ECType */ AAX_CTypeID iCurveType,
00900 float *oXMin, float *oXMax, float *oYMin, float *oYMax ) const = 0;
00901 };
00902
00903 class AAX_IACFEffEffectParameters_V4 : public AAX_IACFEffEffectParameters_V3
00904 {
00905 public:
00906     virtual AAX_Result      UpdatePageTable(uint32_t inTableType, int32_t inTablePageSize, IACFUnknown*
00907 iHostUnknown, IACFUnknown* ioPageTableUnknown) const = 0;
00908 };
00909
00910 #ifdef __clang__
00911 #pragma clang diagnostic pop
00912 #endif
00913
00914 #endif // AAX_IACFEFFECTPARAMETERS_H

```

15.158 AAX_IACFFeatureInfo.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFFeatureInfo](#)

15.159 AAX_IACFFeatureInfo.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00011 #ifndef AAXLibrary_AAX_IACFFeatureInfo_h
00012 #define AAXLibrary_AAX_IACFFeatureInfo_h
00013
00014 #include "AAX.h"
00015

```



```

00016 class AAX_IACFPropertyMap;
00017
00018 #ifdef __clang__
00019 #pragma clang diagnostic push
00020 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00021 #endif
00022
00023 #include "acfunknown.h"
00024
00040 class AAX_IACFFeatureInfo : public IACFUnknown
00041 {
00042 public:
00043     // NOTE: Documentation is copied directly from AAX_IFeatureInfo despite an adaptation of parameter
00049     types (AAX_ESupportLevel* to AAX_ESupportLevel&)
00050     virtual AAX_Result SupportLevel(AAX_ESupportLevel* oSupportLevel) const = 0;
00051     // NOTE: Documentation is not copied directly from AAX_IFeatureInfo due to an adaptation of
00059     parameter types (IACFUnknown to AAX_IPropertyMap)
00060     virtual AAX_Result AcquireProperties(IACFUnknown** outProperties) = 0;
00061 };
00062 #ifdef __clang__
00063 #pragma clang diagnostic pop
00064 #endif
00065
00066 #endif
00067 #endif

```

15.160 AAX_IACFHostProcessor.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

15.160.1 Description

The host processor interface that is exposed to the host application.

Classes

- class [AAX_IACFHostProcessor](#)
Versioned interface for an AAX host processing component.
- class [AAX_IACFHostProcessor_V2](#)
Supplemental interface for an AAX host processing component.

15.161 AAX_IACFHostProcessor.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013–2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021

```

```

00022 #ifndef AAX_IACFHOSTPROCESSOR_H
00023 #define AAX_IACFHOSTPROCESSOR_H
00024
00025 #include "AAX.h"
00026
00027 #ifdef __clang__
00028 #pragma clang diagnostic push
00029 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00030 #endif
00031
00032 #include "acfunknown.h"
00033
00034 class AAX_IHostProcessorDelegate;
00035 class AAX_IEffectParameters;
00036 class AAX_IString;
00037
00051 class AAX_IACFHostProcessor : public IACFUnknown
00052 {
00053 public:
00054     virtual AAX_Result      Initialize(IACFUnknown* iController) = 0;
00055     virtual AAX_Result      Uninitialize() = 0;
00056
00057     virtual AAX_Result      InitOutputBounds ( int64_t iSrcStart, int64_t iSrcEnd, int64_t *
oDstStart, int64_t * oDstEnd ) = 0;
00058     virtual AAX_Result      SetLocation ( int64_t iSample ) = 0;
00059
00060     virtual AAX_Result      RenderAudio ( const float * const inAudioIns [], int32_t inAudioInCount,
float * const iAudioOuts [], int32_t iAudioOutCount, int32_t * ioWindowSize ) = 0;
00061     virtual AAX_Result      PreRender ( int32_t inAudioInCount, int32_t iAudioOutCount, int32_t
iWindowSize ) = 0;
00062     virtual AAX_Result      PostRender () = 0;
00063
00064     virtual AAX_Result      AnalyzeAudio ( const float * const inAudioIns [], int32_t
inAudioInCount, int32_t * ioWindowSize ) = 0;
00065     virtual AAX_Result      PreAnalyze ( int32_t inAudioInCount, int32_t iWindowSize ) = 0;
00066     virtual AAX_Result      PostAnalyze () = 0;
00067 };
00068
00076 class AAX_IACFHostProcessor_V2 : public AAX_IACFHostProcessor
00077 {
00078 public:
00079     virtual AAX_Result      GetClipNameSuffix ( int32_t inMaxLength, AAX_IString* outString ) const
= 0;
00080 };
00081
00082 #ifdef __clang__
00083 #pragma clang diagnostic pop
00084 #endif
00085
00086 #endif //AAX_IACFHOSTPROCESSOR_H

```

15.162 AAX_IACFHostProcessorDelegate.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFHostProcessorDelegate](#)
Versioned interface for host methods specific to offline processing.
- class [AAX_IACFHostProcessorDelegate_V2](#)
Versioned interface for host methods specific to offline processing.
- class [AAX_IACFHostProcessorDelegate_V3](#)
Versioned interface for host methods specific to offline processing.

15.163 AAX_IACFHostProcessorDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011 */
00012
00017 /*=====*/
00018
00019
00020 #ifndef AAX_IACFHOSTPROCESSORDELEGATE_H
00021 #define AAX_IACFHOSTPROCESSORDELEGATE_H
00022
00023 #include "AAX.h"
00024
00025 #ifdef __clang__
00026 #pragma clang diagnostic push
00027 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00028 #endif
00029
00030 #include "acfunknown.h"
00031
00032
00035 class AAX_IACFHostProcessorDelegate : public IACFUnknown
00036 {
00037 public:
00038     virtual AAX_Result    GetAudio ( const float * const inAudioIns [], int32_t inAudioInCount,
00039                                     int64_t inLocation, int32_t * ioNumSamples ) = 0;
00039     virtual int32_t        GetSideChainInputNum () = 0;
00040 };
00041
00042
00045 class AAX_IACFHostProcessorDelegate_V2 : public AAX_IACFHostProcessorDelegate
00046 {
00047 public:
00048     virtual AAX_Result    ForceAnalyze () = 0;
00049 };
00050
00053 class AAX_IACFHostProcessorDelegate_V3 : public AAX_IACFHostProcessorDelegate_V2
00054 {
00055 public:
00056     virtual AAX_Result    ForceProcess () = 0;
00057 };
00058
00059 #ifdef __clang__
00060 #pragma clang diagnostic pop
00061 #endif
00062
00063 #endif // #ifndef AAX_IACFHOSTPROCESSORDELEGATE_H

```

15.164 AAX_IACFHostServices.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFHostServices](#)
Versioned interface to diagnostic and debugging services provided by the AAX host.
- class [AAX_IACFHostServices_V2](#)
V2 of versioned interface to diagnostic and debugging services provided by the AAX host.
- class [AAX_IACFHostServices_V3](#)
V3 of versioned interface to diagnostic and debugging services provided by the AAX host.

15.165 AAX_IACFHostServices.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2018, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00017  /*=====*/
00018
00019
00020  #ifndef AAX_IACFHOSTSERVICES_H
00021  #define AAX_IACFHOSTSERVICES_H
00022
00023  #include "AAX.h"
00024
00025  #ifdef __clang__
00026  #pragma clang diagnostic push
00027  #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00028  #endif
00029
00030  #include "acfunknown.h"
00031
00034  class AAX_IACFHostServices : public IACFUnknown
00035  {
00036  public:
00056      virtual AAX_Result Assert ( const char * iFile, int32_t iLine, const char * iNote ) = 0;
00057
00058      virtual AAX_Result Trace ( int32_t iPriority, const char * iMessage ) = 0;
00059  };
00060
00063  class AAX_IACFHostServices_V2 : public AAX_IACFHostServices
00064  {
00065  public:
00066      virtual AAX_Result StackTrace ( int32_t iTracePriority, int32_t iStackTracePriority, const char *
iMessage ) = 0;
00067  };
00068
00071  class AAX_IACFHostServices_V3 : public AAX_IACFHostServices_V2
00072  {
00073  public:
00074      virtual AAX_Result HandleAssertFailure ( const char * iFile, int32_t iLine, const char * iNote, /*
AAX_EAssertFlags */ int32_t iFlags ) const = 0;
00075  };
00076
00077  #ifdef __clang__
00078  #pragma clang diagnostic pop
00079  #endif
00080
00081  #endif // #ifndef AAX_IACFHOSTSERVICES_H

```

15.166 AAX_IACFPageTable.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFPageTable](#)
Versioned interface to the host's representation of a plug-in instance's page table.
- class [AAX_IACFPageTable_V2](#)
Versioned interface to the host's representation of a plug-in instance's page table.

15.167 AAX_IACFPageTable.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00011 #ifndef AAXLibrary_AAX_IACFPageTable_h
00012 #define AAXLibrary_AAX_IACFPageTable_h
00013
00014 // AAX Includes
00015 #include "AAX.h"
00016
00017 #ifdef __clang__
00018 #pragma clang diagnostic push
00019 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00020 #endif
00021
00022 // ACF Includes
00023 #include "acfunknown.h"
00024
00025 // Forward declarations
00026 class AAX_IString;
00027
00030 class AAX_IACFPageTable : public IACFUnknown
00031 {
00032 public:
00033     virtual AAX_Result Clear() = 0;
00034     virtual AAX_Result Empty(AAX_CBoolean& oEmpty) const = 0;
00035     virtual AAX_Result GetNumPages(int32_t& oNumPages) const = 0;
00036     virtual AAX_Result InsertPage(int32_t iPage) = 0;
00037     virtual AAX_Result RemovePage(int32_t iPage) = 0;
00038     virtual AAX_Result GetNumMappedParameterIDs(int32_t iPage, int32_t& oNumParameterIdentifiers)
00039     const = 0;
00039     virtual AAX_Result ClearMappedParameter(int32_t iPage, int32_t iIndex) = 0;
00040     virtual AAX_Result GetMappedParameterID(int32_t iPage, int32_t iIndex, AAX_IString&
00041     oParameterIdentifier) const = 0;
00041     virtual AAX_Result MapParameterID(AAX_CParamID iParameterIdentifier, int32_t iPage, int32_t
00042     iIndex) = 0;
00043 };
00043
00046 class AAX_IACFPageTable_V2 : public AAX_IACFPageTable
00047 {
00048 public:
00049     virtual AAX_Result GetNumParametersWithNameVariations(int32_t& oNumParameterIdentifiers) const =
00050     0;
00050     virtual AAX_Result GetNameVariationParameterIDAtIndex(int32_t iIndex, AAX_IString&
00051     oParameterIdentifier) const = 0;
00051     virtual AAX_Result GetNumNameVariationsForParameter(AAX_CPageTableParamID iParameterIdentifier,
00052     int32_t& oNumVariations) const = 0;
00052     virtual AAX_Result GetParameterNameVariationAtIndex(AAX_CPageTableParamID iParameterIdentifier,
00053     int32_t iIndex, AAX_IString& oNameVariation, int32_t& oLength) const = 0;
00053     virtual AAX_Result GetParameterNameVariationOfLength(AAX_CPageTableParamID iParameterIdentifier,
00054     int32_t iLength, AAX_IString& oNameVariation) const = 0;
00054     virtual AAX_Result ClearParameterNameVariations() = 0;
00055     virtual AAX_Result ClearNameVariationsForParameter(AAX_CPageTableParamID iParameterIdentifier) =
00056     0;
00056     virtual AAX_Result SetParameterNameVariation(AAX_CPageTableParamID iParameterIdentifier, const
00057     AAX_IString& iNameVariation, int32_t iLength) = 0;
00057 };
00058
00059 #ifdef __clang__
00060 #pragma clang diagnostic pop
00061 #endif
00062
00063
00064 #endif // AAXLibrary_AAX_IACFPageTable_h

```

15.168 AAX_IACFPageTableController.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFPageTableController](#)
Interface for host operations related to the page tables for this plug-in.
- class [AAX_IACFPageTableController_V2](#)
Interface for host operations related to the page tables for this plug-in.

15.169 AAX_IACFPageTableController.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00011 #ifndef AAXLibrary_AAX_IACFPageTableController_h
00012 #define AAXLibrary_AAX_IACFPageTableController_h
00013
00014 #include "AAX.h"
00015
00016 #ifdef __clang__
00017 #pragma clang diagnostic push
00018 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00019 #endif
00020
00021 #include "acfunknown.h"
00022
00023
00024 class AAX_IACFPageTableController : public IACFUnknown
00025 {
00026 public:
00027     // NOTE: Documentation is not copied directly from AAX_IController due to an adaptation of
00028     parameter types (IACFUnknown to AAX_IPageTable)
00029     virtual
00030     AAX_Result
00031     CopyTableForEffect(
00032         AAX_CPropertyValue inManufacturerID,
00033         AAX_CPropertyValue inProductID,
00034         AAX_CPropertyValue inPlugInID,
00035         uint32_t inTableType,
00036         int32_t inTablePageSize,
00037         IACFUnknown* oPageTable) const = 0;
00038
00039     // NOTE: Documentation is not copied directly from AAX_IController due to an adaptation of
00040     parameter types (IACFUnknown to AAX_IPageTable)
00041     virtual
00042     AAX_Result
00043     CopyTableOfLayoutForEffect(
00044         const char * inEffectID,
00045         const char * inLayoutName,
00046         uint32_t inTableType,
00047         int32_t inTablePageSize,
00048         IACFUnknown* oPageTable) const = 0;
00049 };
00050
00051 class AAX_IACFPageTableController_V2 : public AAX_IACFPageTableController
00052 {
00053 public:
00054     virtual
00055     AAX_Result
00056     CopyTableForEffectFromFile(
00057         const char* inPageTableFilePath,
00058         AAX_ETextEncoding inFilePathEncoding,
00059         AAX_CPropertyValue inManufacturerID,
00060         AAX_CPropertyValue inProductID,
00061         AAX_CPropertyValue inPlugInID,
00062         uint32_t inTableType,
00063         int32_t inTablePageSize,
00064         IACFUnknown* oPageTable) const = 0;
00065
00066     virtual
00067     AAX_Result
00068     CopyTableOfLayoutFromFile(

```

```

00185         const char* inPageTableFilePath,
00186         AAX_ETextEncoding inFilePathEncoding,
00187         const char* inLayoutName,
00188         uint32_t inTableType,
00189         int32_t inTablePageSize,
00190         IACFUnknown* oPageTable) const = 0;
00191 };
00192
00193 #ifdef __clang__
00194 #pragma clang diagnostic pop
00195 #endif
00196
00197 #endif

```

15.170 AAX_IACFPrivateDataAccess.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

15.170.1 Description

Interface for the AAX host's data access functionality.

Classes

- class [AAX_IACFPrivateDataAccess](#)
Interface for the AAX host's data access functionality.

15.171 AAX_IACFPrivateDataAccess.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef _AAX_IACFPrivateDATAACCESS_H_
00023 #define _AAX_IACFPrivateDATAACCESS_H_
00024
00025 #include "AAX.h"
00026
00027 #ifdef __clang__
00028 #pragma clang diagnostic push
00029 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00030 #endif
00031
00032 #include "acfunknown.h"
00033
00039 class AAX_IACFPrivateDataAccess : public IACFUnknown
00040 {
00041 public:
00042

```

```

00043     virtual AAX_Result    ReadPortDirect( AAX_CFieldIndex inFieldIndex, const uint32_t inOffset, const
uint32_t inSize, void* outBuffer ) = 0;
00044     virtual AAX_Result    WritePortDirect( AAX_CFieldIndex inFieldIndex, const uint32_t inOffset,
const uint32_t inSize, const void* inBuffer ) = 0;
00045
00046 };
00047
00048 #ifdef __clang__
00049 #pragma clang diagnostic pop
00050 #endif
00051
00052 #endif // #ifndef _AAX_IACFPrivateDATAACCESS_H_

```

15.172 AAX_IACFPropertyMap.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

15.172.1 Description

Versioned interface for an [AAX_IPropertyMap](#).

Classes

- class [AAX_IACFPropertyMap](#)
Versioned interface for an [AAX_IPropertyMap](#).
- class [AAX_IACFPropertyMap_V2](#)
Versioned interface for an [AAX_IPropertyMap](#).
- class [AAX_IACFPropertyMap_V3](#)
Versioned interface for an [AAX_IPropertyMap](#).

15.173 AAX_IACFPropertyMap.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IACFPROPERTYMAP_H
00023 #define AAX_IACFPROPERTYMAP_H
00024
00025 #include "AAX.h"
00026
00027 #ifdef __clang__
00028 #pragma clang diagnostic push
00029 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00030 #endif
00031
00032 #include "acfunknown.h"

```



```

00033
00036 class AAX_IACFPropertyMap : public IACFUnknown
00037 {
00038 public:
00039     virtual AAX_CBoolean      GetProperty ( AAX_EProperty inProperty, AAX_CPropertyValue * outValue
                                ) const = 0;
00040     virtual AAX_Result        AddProperty ( AAX_EProperty inProperty, AAX_CPropertyValue inValue )
                                = 0;
00041     virtual AAX_Result        RemoveProperty ( AAX_EProperty inProperty ) = 0;
00042 };
00043
00046 class AAX_IACFPropertyMap_V2 : public AAX_IACFPropertyMap
00047 {
00048 public:
00049     virtual AAX_Result        AddPropertyWithIDArray ( AAX_EProperty inProperty, const
AAX_SPlugInIdentifierTriad* inPluginIDs, uint32_t inNumPluginIDs) = 0;
00050     virtual AAX_CBoolean      GetPropertyWithIDArray ( AAX_EProperty inProperty, const
AAX_SPlugInIdentifierTriad** outPluginIDs, uint32_t* outNumPluginIDs) const = 0;
00051 };
00052
00055 class AAX_IACFPropertyMap_V3 : public AAX_IACFPropertyMap_V2
00056 {
00057 public:
00058     virtual AAX_CBoolean      GetProperty64 ( AAX_EProperty inProperty, AAX_CPropertyValue64 *
outValue ) const = 0;
00059     virtual AAX_Result        AddProperty64 ( AAX_EProperty inProperty, AAX_CPropertyValue64
inValue ) = 0;
00060 };
00061
00062 #ifdef __clang__
00063 #pragma clang diagnostic pop
00064 #endif
00065
00066 #endif // AAX_IACFPROPERTYMAP_H

```

15.174 AAX_IACFSessionDocument.h File Reference

```

#include "AAX_UIDs.h"
#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFSessionDocument](#)
Interface representing information in a host session document.

Macros

- #define [AAX_IACFSessionDocument_H](#)

15.174.1 Macro Definition Documentation

15.174.1.1 AAX_IACFSessionDocument_H

```
#define AAX_IACFSessionDocument_H
```

15.175 AAX_IACFSessionDocument.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011 */
00012
00016 /*=====*/
00017
00018 #pragma once
00019 #ifndef AAX_IACFSessionDocument_H
00020 #define AAX_IACFSessionDocument_H
00021
00022 #ifdef __clang__
00023 #pragma clang diagnostic push
00024 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00025 #endif
00026
00027 #include "AAX_UIDs.h"
00028 #include "AAX.h"
00029 #include "acfunknown.h"
00030
00037 class AAX_IACFSessionDocument : public IACFUnknown
00038 {
00039 public:
00045     virtual AAX_Result GetDocumentData(AAX_DocumentData_UID const & inDataType, IACFUnknown **
        outData) = 0;
00046 };
00047
00048 #ifdef __clang__
00049 #pragma clang diagnostic pop
00050 #endif
00051
00052 #endif // AAX_IACFSessionDocument_H

```

15.176 AAX_IACFSessionDocumentClient.h File Reference

```

#include "AAX_UIDs.h"
#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFSessionDocumentClient](#)
Interface representing a client of the session document interface.

Macros

- #define [AAX_IACFSessionDocumentClient_H](#)

15.176.1 Macro Definition Documentation

15.176.1.1 AAX_IACFSessionDocumentClient_H

```
#define AAX_IACFSessionDocumentClient_H
```

15.177 AAX_IACFSessionDocumentClient.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011 */
00012
00016 /*=====*/
00017
00018 #pragma once
00019 #ifndef AAX_IACFSessionDocumentClient_H
00020 #define AAX_IACFSessionDocumentClient_H
00021
00022 #ifdef __clang__
00023 #pragma clang diagnostic push
00024 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00025 #endif
00026
00027 #include "AAX_UIDs.h"
00028 #include "AAX.h"
00029 #include "acfunknown.h"
00030
00037 class AAX_IACFSessionDocumentClient : public IACFUnknown
00038 {
00039 public:
00043     virtual AAX_Result Initialize (IACFUnknown * iUnknown) = 0;
00044     virtual AAX_Result Uninitialize (void) = 0;
00046
00058     virtual AAX_Result SetSessionDocument (IACFUnknown * iSessionDocument) = 0;
00060
00088     virtual AAX_Result NotificationReceived(/* AAX_ENotificationEvent */ AAX_CTypeID
inNotificationType, const void * inNotificationData, uint32_t inNotificationDataSize) = 0;
00090 };
00091
00092 #ifdef __clang__
00093 #pragma clang diagnostic pop
00094 #endif
00095
00096 #endif // AAX_IACFSessionDocumentClient_H
```

15.178 AAX_IACFTask.h File Reference

```
#include "AAX.h"
#include "acfunknown.h"
```

15.178.1 Description

Defines the interface representing an asynchronous task.

Classes

- class [AAX_IACFTask](#)
Versioned interface for an asynchronous task.

Macros

- `#define AAX_IACFTask_H`

Enumerations

- enum class `AAX_TaskCompletionStatus` : `int32_t` {
`AAX_TaskCompletionStatus::None` = 0 ,
`AAX_TaskCompletionStatus::Done` = 1 ,
`AAX_TaskCompletionStatus::Canceled` = 2 ,
`AAX_TaskCompletionStatus::Error` = 3 }

15.178.2 Macro Definition Documentation

15.178.2.1 AAX_IACFTask_H

```
#define AAX_IACFTask_H
```

15.179 AAX_IACFTask.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020
00021 #pragma once
00022
00023 #ifndef AAX_IACFTask_H
00024 #define AAX_IACFTask_H
00025
00026 #include "AAX.h"
00027
00028 #ifdef __clang__
00029 #pragma clang diagnostic push
00030 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00031 #endif
00032
00033 #include "acfunknown.h"
00034
00035 class AAX_IACFDataBuffer;
00036
00042 enum class AAX_TaskCompletionStatus : int32_t {
00043     None = 0
00044     ,Done = 1
00045     ,Canceled = 2
00046     ,Error = 3
00047 };
00048
00056 class AAX_IACFTask : public IACFUnknown
00057 {
00058 public:
00059     virtual AAX_Result GetType(AAX_CTypeID * oType) const = 0;
00060     virtual AAX_IACFDataBuffer const * GetArgumentOfType(AAX_CTypeID iType) const = 0;
```

```

00061
00062     virtual AAX_Result SetProgress(float iProgress) = 0;
00063     virtual float GetProgress() const = 0;
00064
00065     virtual AAX_Result AddResult(AAX_IACFDataBuffer const * iResult) = 0;
00066
00078     virtual AAX_Result SetDone(AAX_TaskCompletionStatus iStatus) = 0;
00079 };
00080
00081 #ifdef __clang__
00082 #pragma clang diagnostic pop
00083 #endif
00084
00085 #endif //AAX_IACFTask_H

```

15.180 AAX_IACFTaskAgent.h File Reference

```

#include "AAX.h"
#include "acfunknown.h"

```

Classes

- class [AAX_IACFTaskAgent](#)
Versioned interface for a component that accepts task requests.

Macros

- #define [AAX_IACFTaskAgent_H](#)

15.180.1 Macro Definition Documentation

15.180.1.1 AAX_IACFTaskAgent_H

```
#define AAX_IACFTaskAgent_H
```

15.181 AAX_IACFTaskAgent.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once

```

```

00019
00020 #ifndef AAX_IACFTaskAgent_H
00021 #define AAX_IACFTaskAgent_H
00022
00023 #include "AAX.h"
00024
00025 #ifdef __clang__
00026 #pragma clang diagnostic push
00027 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00028 #endif
00029
00030 #include "acfunknown.h"
00031
00032 class IACFUnknown;
00033
00034
00049 class AAX_IACFTaskAgent : public IACFUnknown
00050 {
00051 public:
00062     virtual AAX_Result Initialize(IACFUnknown* iController) = 0;
00068     virtual AAX_Result Uninitialize() = 0;
00070
00082     virtual AAX_Result AddTask(IACFUnknown * iTask) = 0;
00086     virtual AAX_Result CancelAllTasks() = 0;
00088 };
00089
00090
00091 #ifdef __clang__
00092 #pragma clang diagnostic pop
00093 #endif
00094
00095 #endif

```

15.182 AAX_IACFTransport.h File Reference

```

#include "AAX.h"
#include "AAX_Enums.h"
#include "acfunknown.h"

```

15.182.1 Description

Interface for accessing the host's transport state.

Classes

- class [AAX_IACFTransport](#)
Versioned interface to get information about the host's transport state.
- class [AAX_IACFTransport_V2](#)
Versioned interface to get information about the host's transport state.
- class [AAX_IACFTransport_V3](#)
Versioned interface to get information about the host's transport state.
- class [AAX_IACFTransport_V4](#)
Versioned interface to get information about the host's transport state.

15.183 AAX_IACFTransport.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2019-2021, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012
00013  /*=====*/
00014
00015  #ifndef AAX_IACFTRANSPORT_H
00016  #define AAX_IACFTRANSPORT_H
00017
00018  #pragma once
00019
00020  #include "AAX.h"
00021  #include "AAX_Enums.h"
00022
00023  #ifdef __clang__
00024  #pragma clang diagnostic push
00025  #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00026  #endif
00027
00028  #include "acfunknown.h"
00029
00030  class AAX_IACFTransport : public IACFUnknown
00031  {
00032  public:
00033
00034      virtual AAX_Result    GetCurrentTempo ( double* TempoBPM ) const = 0;
00035      virtual AAX_Result    GetCurrentMeter ( int32_t* MeterNumerator, int32_t* MeterDenominator ) const
00036      = 0;
00037      virtual AAX_Result    IsTransportPlaying ( bool* isPlaying ) const = 0;
00038      virtual AAX_Result    GetCurrentTickPosition ( int64_t* TickPosition ) const = 0;
00039      virtual AAX_Result    GetCurrentLoopPosition ( bool* bLooping, int64_t* LoopStartTick, int64_t*
00040      LoopEndTick ) const = 0;
00041      virtual AAX_Result    GetCurrentNativeSampleLocation ( int64_t* SampleLocation ) const = 0;
00042      virtual AAX_Result    GetCustomTickPosition ( int64_t* oTickPosition, int64_t iSampleLocation )
00043      const = 0;
00044      virtual AAX_Result    GetBarBeatPosition ( int32_t* Bars, int32_t* Beats, int64_t* DisplayTicks,
00045      int64_t SampleLocation ) const = 0;
00046      virtual AAX_Result    GetTicksPerQuarter ( uint32_t* ticks ) const = 0;
00047      virtual AAX_Result    GetCurrentTicksPerBeat ( uint32_t* ticks ) const = 0;
00048
00049  };
00050
00051  class AAX_IACFTransport_V2 : public AAX_IACFTransport
00052  {
00053  public:
00054
00055      virtual AAX_Result    GetTimelineSelectionStartPosition( int64_t* oSampleLocation ) const = 0;
00056      virtual AAX_Result    GetTimeCodeInfo( AAX_EFrameRate* oFrameRate, int32_t* oOffset ) const = 0;
00057      virtual AAX_Result    GetFeetFramesInfo( AAX_EFeetFramesRate* oFeetFramesRate, int64_t* oOffset )
00058      const = 0;
00059      virtual AAX_Result    IsMetronomeEnabled ( int32_t* isEnabled ) const = 0;
00060
00061  };
00062
00063  class AAX_IACFTransport_V3 : public AAX_IACFTransport_V2
00064  {
00065  public:
00066
00067      virtual AAX_Result    GetHDDTimeCodeInfo( AAX_EFrameRate* oHDDFrameRate, int64_t* oHDDOffset ) const
00068      = 0;
00069
00070  };
00071
00072  class AAX_IACFTransport_V4 : public AAX_IACFTransport_V3
00073  {
00074  public:
00075
00076      virtual AAX_Result    GetTimelineSelectionEndPosition( int64_t* oSampleLocation ) const = 0;
00077
00078  };
00079
00080  #ifdef __clang__
00081  #pragma clang diagnostic pop
00082  #endif
00083
00084  #endif // AAX_IACFTRANSPORT_H
00085

```

15.184 AAX_IACFTransportControl.h File Reference

```
#include "AAX.h"
#include "AAX_Enums.h"
#include "acfunknown.h"
```

15.184.1 Description

Interface for control over the host's transport state.

Classes

- class [AAX_IACFTransportControl](#)
Versioned interface to control the host's transport state.

15.185 AAX_IACFTransportControl.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00013
00014 #ifndef AAX_IACFTRANSPORTCONTROL_H
00015 #define AAX_IACFTRANSPORTCONTROL_H
00016
00017 #pragma once
00018
00019 #include "AAX.h"
00020 #include "AAX_Enums.h"
00021
00022 #ifdef __clang__
00023 #pragma clang diagnostic push
00024 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00025 #endif
00026 #include "acfunknown.h"
00027
00028 class AAX_IACFTransportControl : public IACFUnknown
00029 {
00030 public:
00031     virtual AAX_Result RequestTransportStart() = 0;
00032     virtual AAX_Result RequestTransportStop() = 0;
00033 };
00034
00035 #ifdef __clang__
00036 #pragma clang diagnostic pop
00037 #endif
00038 #endif // AAX_IACFTRANSPORTCONTROL_H
00039
```


15.186 AAX_IACFViewContainer.h File Reference

```
#include "AAX_GUITypes.h"
#include "AAX.h"
#include "acfunknown.h"
```

15.186.1 Description

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAXHost and by effect components.

Classes

- class [AAX_IACFViewContainer](#)
Interface for the AAX host's view of a single instance of an effect. Used by both clients of the host app and by effect components.
- class [AAX_IACFViewContainer_V2](#)
Supplemental interface for the AAX host's view of a single instance of an effect. Used by both clients of the host app and by effect components.
- class [AAX_IACFViewContainer_V3](#)
Additional methods to track mouse as it moves over controls.

15.187 AAX_IACFViewContainer.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00020 /*=====*/
00021
00022
00023 #ifndef _AAX_IACFVIEWCONTAINER_H_
00024 #define _AAX_IACFVIEWCONTAINER_H_
00025
00026 #include "AAX_GUITypes.h"
00027 #include "AAX.h"
00028
00029 #ifdef __clang__
00030 #pragma clang diagnostic push
00031 #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00032 #endif
00033
00034 #include "acfunknown.h"
00035
00042 class AAX_IACFViewContainer : public IACFUnknown
00043 {
00044 public:
00048     virtual int32_t      GetType () = 0;
00049     virtual void *       GetPtr () = 0;
00050     virtual AAX_Result   GetModifiers ( uint32_t * outModifiers ) = 0;
00052
00056     virtual AAX_Result   SetViewSize ( AAX_Point & inSize ) = 0;
00058
```

```

00062     virtual AAX_Result    HandleParameterMouseDown    ( AAX_CParamID inParamID, uint32_t inModifiers )
= 0;
00063     virtual AAX_Result    HandleParameterMouseDrag    ( AAX_CParamID inParamID, uint32_t inModifiers )
= 0;
00064     virtual AAX_Result    HandleParameterMouseUp      ( AAX_CParamID inParamID, uint32_t inModifiers
) = 0;
00066 };
00067
00068
00075 class AAX_IACFViewContainer_V2 : public AAX_IACFViewContainer
00076 {
00077 public:
00081     virtual AAX_Result    HandleMultipleParametersMouseDown ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) = 0;
00082     virtual AAX_Result    HandleMultipleParametersMouseDrag ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) = 0;
00083     virtual AAX_Result    HandleMultipleParametersMouseUp   ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) = 0;
00085 };
00086
00087
00093 class AAX_IACFViewContainer_V3 : public AAX_IACFViewContainer_V2
00094 {
00095 public:
00099     virtual AAX_Result    HandleParameterMouseEnter(AAX_CParamID inParamID, uint32_t inModifiers ) =
0;
00100     virtual AAX_Result    HandleParameterMouseExit(AAX_CParamID inParamID, uint32_t inModifiers ) = 0;
00102 };
00103
00104
00105 #ifdef __clang__
00106 #pragma clang diagnostic pop
00107 #endif
00108
00109 #endif

```

15.188 AAX_IAutomationDelegate.h File Reference

```
#include "AAX.h"
```

15.188.1 Description

Interface allowing an AAX plug-in to interact with the host's automation system.

Classes

- class [AAX_IAutomationDelegate](#)

Interface allowing an AAX plug-in to interact with the host's event system.

15.189 AAX_IAutomationDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */

```

```

00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IAUTOMATIONDELEGATE_H
00023 #define AAX_IAUTOMATIONDELEGATE_H
00024
00025 #include "AAX.h"
00026
00043 class AAX_IAutomationDelegate
00044 {
00045 public:
00046
00047     virtual ~AAX_IAutomationDelegate() {}
00048
00061     virtual AAX_Result    RegisterParameter ( AAX_CParamID iParameterID ) = 0;
00062
00073     virtual AAX_Result    UnregisterParameter ( AAX_CParamID iParameterID ) = 0;
00074
00083     virtual AAX_Result    PostSetValueRequest ( AAX_CParamID iParameterID, double normalizedValue )
00084     const = 0;
00093     virtual AAX_Result    PostCurrentValue( AAX_CParamID iParameterID, double normalizedValue ) const
00094     = 0;
00101     virtual AAX_Result    PostTouchRequest( AAX_CParamID iParameterID ) = 0;
00102
00109     virtual AAX_Result    PostReleaseRequest( AAX_CParamID iParameterID ) = 0;
00110
00119     virtual AAX_Result    GetTouchState ( AAX_CParamID iParameterID, AAX_CBoolean * oTouched ) = 0;
00120
00131     virtual AAX_Result    ParameterNameChanged ( AAX_CParamID iParameterID ) = 0;
00132 };
00133
00134
00135 #endif

```

15.190 AAX_ICollection.h File Reference

```
#include "AAX.h"
```

15.190.1 Description

Interface to represent a plug-in binary's static description.

Classes

- class [AAX_ICollection](#)
Interface to represent a plug-in binary's static description.

15.191 AAX_ICollection.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */

```

```

00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_ICOLLECTION_H
00023 #define AAX_ICOLLECTION_H
00024
00025 #include "AAX.h"
00026
00027 class AAX_IEffectDescriptor;
00028 class AAX_IPropertyMap;
00029 class AAX_IDescriptionHost;
00030 class IACFDefinition;
00031
00050 class AAX_ICollection
00051 {
00052 public:
00053     virtual ~AAX_ICollection() {}
00054
00055 public: // AAX_IACFCollection
00056
00060     virtual AAX_IEffectDescriptor *      NewDescriptor () = 0;
00061
00083     virtual AAX_Result                    AddEffect ( const char * inEffectID, AAX_IEffectDescriptor*
inEffectDescriptor ) = 0;
00084
00091     virtual AAX_Result                    SetManufacturerName( const char* inPackageName ) = 0;
00103     virtual AAX_Result                    AddPackageName( const char *inPackageName ) = 0;
00110     virtual AAX_Result                    SetPackageVersion( uint32_t inVersion ) = 0;
00115     virtual AAX_IPropertyMap *            NewPropertyMap () = 0;
00123     virtual AAX_Result                    SetProperties ( AAX_IPropertyMap * inProperties ) =
0;
00124
00125 public: // AAX_ICollection
00126
00134     virtual AAX_IDescriptionHost* DescriptionHost() = 0;
00135     virtual const AAX_IDescriptionHost* DescriptionHost() const = 0;
00136
00148     virtual IACFDefinition* HostDefinition() const = 0;
00149
00150 };
00151
00152 #endif

```

15.192 AAX_IComponentDescriptor.h File Reference

```

#include "AAX.h"
#include "AAX_IDma.h"
#include "AAX_Callbacks.h"

```

15.192.1 Description

Description interface for an AAX plug-in algorithm.

Classes

- class [AAX_IComponentDescriptor](#)
Description interface for an AAX plug-in component.

15.193 AAX_IComponentDescriptor.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00019  /*=====*/
00020
00021
00022  #ifndef _AAX_ICOMPONENTDESCRIPTOR_H_
00023  #define _AAX_ICOMPONENTDESCRIPTOR_H_
00024
00025  #include "AAX.h"
00026  #include "AAX_IDma.h"
00027  #include "AAX_Callbacks.h"
00028
00029  class AAX_IPropertyMap;
00030
00031
00043  class AAX_IComponentDescriptor
00044  {
00045  public:
00046      virtual ~AAX_IComponentDescriptor() {}
00047
00054      virtual AAX_Result          Clear () = 0;
00055
00068      virtual AAX_Result          AddAudioIn ( AAX_CFieldIndex inFieldIndex ) = 0;
00069
00082      virtual AAX_Result          AddAudioOut ( AAX_CFieldIndex inFieldIndex ) = 0;
00083
00096      virtual AAX_Result          AddAudioBufferLength ( AAX_CFieldIndex inFieldIndex ) = 0;
00097
00110      virtual AAX_Result          AddSampleRate ( AAX_CFieldIndex inFieldIndex ) = 0;
00111
00129      virtual AAX_Result          AddClock ( AAX_CFieldIndex inFieldIndex ) = 0;
00130
00144      virtual AAX_Result          AddSideChainIn ( AAX_CFieldIndex inFieldIndex ) = 0;
00145
00168      virtual AAX_Result          AddDataInPort ( AAX_CFieldIndex inFieldIndex, uint32_t
inPacketSize, AAX_EDataInPortType inPortType = AAX_eDataInPortType_Buffered ) = 0;
00169
00205      virtual AAX_Result          AddAuxOutputStem ( AAX_CFieldIndex inFieldIndex, int32_t
inStemFormat, const char inNameUTF8[]) = 0;
00222      virtual AAX_Result          AddPrivateData ( AAX_CFieldIndex inFieldIndex, int32_t
inDataSize, /* AAX_EPrivateDataOptions */ uint32_t inOptions = AAX_ePrivateDataOptions_DefaultOptions
) = 0;
00223
00238      virtual AAX_Result          AddTemporaryData( AAX_CFieldIndex inFieldIndex, uint32_t
inDataElementSize) = 0;
00239
00260      virtual AAX_Result          AddDmaInstance ( AAX_CFieldIndex inFieldIndex,
AAX_IDma::EMode inDmaMode ) = 0;
00261
00277      virtual AAX_Result          AddMeters ( AAX_CFieldIndex inFieldIndex, const AAX_CTypeID*
inMeterIDs, const uint32_t inMeterCount ) = 0;
00278
00301      virtual AAX_Result          AddMIDINode ( AAX_CFieldIndex inFieldIndex,
AAX_EMIDINodeType inNodeType, const char inNodeName[], uint32_t channelMask ) = 0;
00302
00313      virtual AAX_Result          AddReservedField ( AAX_CFieldIndex inFieldIndex, uint32_t
inFieldType ) = 0;
00314
00322      virtual AAX_IPropertyMap *   NewPropertyMap () const = 0; // CONST?
00323
00334      virtual AAX_IPropertyMap *   DuplicatePropertyMap (AAX_IPropertyMap* inPropertyMap) const =
0;
00354      virtual AAX_Result          AddProcessProc_Native (
00355          AAX_CProcessProc inProcessProc,
00356          AAX_IPropertyMap * inProperties = NULL,
00357          AAX_CInstanceInitProc inInstanceInitProc = NULL,
00358          AAX_CBackgroundProc inBackgroundProc = NULL,
00359          AAX_CSelector * outProcID = NULL) = 0;
00383      virtual AAX_Result          AddProcessProc_TI (
00384          const char inDLLFileNameUTF8 [],
00385          const char inProcessProcSymbol [],
00386          AAX_IPropertyMap * inProperties,

```

```

00387     const char inInstanceInitProcSymbol [] = NULL,
00388     const char inBackgroundProcSymbol [] = NULL,
00389     AAX_CSelector * outProcID = NULL) = 0;
00390
00436     virtual AAX_Result      AddProcessProc (
00437         AAX_IPropertyMap* inProperties,
00438         AAX_CSelector* outProcIDs = NULL,
00439         int32_t inProcIDsSize = 0) = 0;
00440
00454     template <typename aContextType>
00455     AAX_Result      AddProcessProc_Native (
00456         void (AAX_CALLBACK *inProcessProc) ( aContextType * const inInstancesBegin [], const void *
inInstancesEnd),
00457         AAX_IPropertyMap * inProperties = NULL,
00458         int32_t (AAX_CALLBACK *inInstanceInitProc) ( const aContextType * inInstanceContextPtr,
AAX_EComponentInstanceInitAction inAction ) = NULL,
00459         int32_t (AAX_CALLBACK *inBackgroundProc) ( void ) = NULL );
00460 };
00461
00462     template <typename aContextType>
00463     inline AAX_Result
00464     AAX_IComponentDescriptor::AddProcessProc_Native (
00465         void (AAX_CALLBACK *inProcessProc) ( aContextType * const inInstancesBegin [], const void *
inInstancesEnd),
00466         AAX_IPropertyMap * inProperties,
00467         int32_t (AAX_CALLBACK *inInstanceInitProc) ( const aContextType * inInstanceContextPtr,
AAX_EComponentInstanceInitAction inAction ),
00468         int32_t (AAX_CALLBACK *inBackgroundProc) ( void ) )
00469     {
00470         return this->AddProcessProc_Native(
00471             reinterpret_cast<AAX_CProcessProc>( inProcessProc ),
00472             inProperties,
00473             reinterpret_cast<AAX_CInstanceInitProc>( inInstanceInitProc ),
00474             reinterpret_cast<AAX_CBackgroundProc>( inBackgroundProc ) );
00475     }
00476
00477 #endif // #ifndef _AAX_ICOMPONENTDESCRIPTOR_H_

```

15.194 AAX_IContainer.h File Reference

15.194.1 Description

Abstract container interface.

Classes

- class [AAX_IContainer](#)

15.195 AAX_IContainer.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2015, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00021 #ifndef AAX_ICONTAINER_H
00022 #define AAX_ICONTAINER_H
00024
00025
00028 class AAX_IContainer

```

```

00029 {
00030 public:
00031     virtual ~AAX_IContainer() {}
00032 public:
00033     enum EStatus
00034     {
00035         eStatus_Success = 0
00036         , eStatus_Overflow = 1
00037         , eStatus_NotInitialized = 2
00038         , eStatus_Unavailable = 3
00039         , eStatus_Unsupported = 4
00040     };
00041 };
00042 public:
00043     virtual void Clear() = 0;
00044 };
00045
00046 #endif /* defined(AAX_ICONTAINER_H) */

```

15.196 AAX_IController.h File Reference

```

#include "AAX_Properties.h"
#include "AAX_IString.h"
#include "AAX.h"
#include <memory>

```

15.196.1 Description

Interface for the AAX host's view of a single instance of an effect.

Classes

- class [AAX_IController](#)

Interface for the AAX host's view of a single instance of an effect. Used by both clients of the AAX host and by effect components.

15.197 AAX_IController.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef _AAX_ICONTROLLER_H_
00023 #define _AAX_ICONTROLLER_H_
00024
00025 #include "AAX_Properties.h"
00026 #include "AAX_IString.h"
00027 #include "AAX.h"
00028 #include <memory>

```

```

00029
00030 // Forward declarations
00031 class AAX_IPageTable;
00032
00033
00041 class AAX_IController
00042 {
00043 public:
00044
00045     virtual ~AAX_IController(void) {}
00046
00052     virtual AAX_Result    GetEffectID (
00053         AAX_IString *    outEffectID) const = 0;
00060     virtual // AAX_VController
00061     AAX_Result
00062     GetSampleRate (
00063         AAX_CSAMPLERate *outSampleRate ) const = 0;
00070     virtual // AAX_VController
00071     AAX_Result
00072     GetInputStemFormat (
00073         AAX_EStemFormat *outStemFormat ) const = 0;
00080     virtual // AAX_VController
00081     AAX_Result
00082     GetOutputStemFormat (
00083         AAX_EStemFormat *outStemFormat) const = 0;
00100     virtual
00101     AAX_Result
00102     GetSignalLatency(
00103         int32_t* outSamples) const = 0;
00127     virtual
00128     AAX_Result
00129     GetCycleCount(
00130         AAX_EProperty inWhichCycleCount,
00131         AAX_CPropertyValue* outNumCycles) const = 0;
00145     virtual
00146     AAX_Result
00147     GetTODLocation (
00148         AAX_CTimeOfDay* outTODLocation ) const = 0;
00150
00180     virtual
00181     AAX_Result
00182     SetSignalLatency(
00183         int32_t inNumSamples) = 0;
00210     virtual
00211     AAX_Result
00212     SetCycleCount(
00213         AAX_EProperty* inWhichCycleCounts,
00214         AAX_CPropertyValue* iValues,
00215         int32_t numValues) = 0;
00217
00248     virtual // AAX_VController
00249     AAX_Result
00250     PostPacket (
00251         AAX_CFieldIndex inFieldIndex,
00252         const void *    inPayloadP,
00253         uint32_t        inPayloadSize) = 0;
00255
00283     virtual AAX_Result    SendNotification (/* AAX_ENotificationEvent */ AAX_CTypeID
inNotificationType, const void* inNotificationData, uint32_t inNotificationDataSize) = 0;
00294     virtual AAX_Result    SendNotification (/* AAX_ENotificationEvent */ AAX_CTypeID
inNotificationType) = 0;
00296
00312     virtual AAX_Result    GetCurrentMeterValue ( AAX_CTypeID inMeterID, float * outMeterValue )
const = 0;
00321     virtual AAX_Result    GetMeterPeakValue ( AAX_CTypeID inMeterID, float * outMeterPeakValue )
const = 0;
00328     virtual AAX_Result    ClearMeterPeakValue ( AAX_CTypeID inMeterID ) const = 0;
00337     virtual AAX_Result    GetMeterCount ( uint32_t * outMeterCount ) const = 0;
00348     virtual AAX_Result    GetMeterClipped ( AAX_CTypeID inMeterID, AAX_CBoolean * outClipped )
const = 0;
00357     virtual AAX_Result    ClearMeterClipped ( AAX_CTypeID inMeterID ) const = 0;
00359
00360
00375     virtual AAX_Result    GetNextMIDIpacket ( AAX_CFieldIndex* outPort, AAX_CMidiPacket* outPacket
) = 0;
00377
00392     virtual
00393     AAX_Result GetHybridSignalLatency(int32_t* outSamples) const = 0;
00408     virtual
00409     AAX_Result GetCurrentAutomationTimestamp(AAX_CTransportCounter* outTimestamp) const = 0;
00419     virtual
00420     AAX_Result GetHostName(AAX_IString* outHostNameString) const = 0;
00427     virtual
00428     AAX_Result GetPlugInTargetPlatform(AAX_CTargetPlatform* outTargetPlatform) const = 0;
00435     virtual
00436     AAX_Result GetIsAudioSuite(AAX_CBoolean* outIsAudioSuite) const = 0;
00437

```



```

00464     virtual
00465     AAX_IPageTable*
00466     CreateTableCopyForEffect(
00467         AAX_CPropertyValue inManufacturerID,
00468         AAX_CPropertyValue inProductID,
00469         AAX_CPropertyValue inPlugInID,
00470         uint32_t inTableType,
00471         int32_t inTablePageSize) const = 0;
00472
00496     virtual
00497     AAX_IPageTable*
00498     CreateTableCopyForLayout(
00499         const char * inEffectID,
00500         const char * inLayoutName,
00501         uint32_t inTableType,
00502         int32_t inTablePageSize) const = 0;
00503
00527     virtual
00528     AAX_IPageTable*
00529     CreateTableCopyForEffectFromFile(
00530         const char* inPageTableFilePath,
00531         AAX_ETextEncoding inFilePathEncoding,
00532         AAX_CPropertyValue inManufacturerID,
00533         AAX_CPropertyValue inProductID,
00534         AAX_CPropertyValue inPlugInID,
00535         uint32_t inTableType,
00536         int32_t inTablePageSize) const = 0;
00537
00556     virtual
00557     AAX_IPageTable*
00558     CreateTableCopyForLayoutFromFile(
00559         const char* inPageTableFilePath,
00560         AAX_ETextEncoding inFilePathEncoding,
00561         const char* inLayoutName,
00562         uint32_t inTableType,
00563         int32_t inTablePageSize) const = 0;
00564 };
00565
00566 #endif // #ifndef _AAX_IPLUGIN_H_

```

15.198 AAX_IDataBuffer.h File Reference

```

#include "AAX_IACFDataBuffer.h"
#include "AAX.h"
#include "CACFUnknown.h"
#include "AAX_UIDs.h"
#include "acfextras.h"

```

Classes

- class [AAX_IDataBuffer](#)
Interface for reference counted data buffers.

Macros

- #define [AAX_IDataBuffer_H](#)

15.198.1 Macro Definition Documentation

15.198.1.1 AAX_IDataBuffer_H

```
#define AAX_IDataBuffer_H
```

15.199 AAX_IDataBuffer.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019
00020 #ifndef AAX_IDataBuffer_H
00021 #define AAX_IDataBuffer_H
00022
00023 #include "AAX_IACFDataBuffer.h"
00024 #include "AAX.h"
00025 #include "CACFUnknown.h"
00026 #include "AAX_UIDs.h"
00027 #include "acfextras.h"
00028
00029
00035 class AAX_IDataBuffer : public AAX_IACFDataBuffer
00036                       , public CACFUnknown
00037 {
00038 public:
00039     ACF_DECLARE_STANDARD_UNKNOWN()
00040
00041     ACFMETHOD(InternalQueryInterface)(const acfIID & riid, void **ppvObjOut) AAX_OVERRIDE
00042     {
00043         if (riid == IID_IAAXDataBufferV1)
00044         {
00045             *ppvObjOut = static_cast<IACFUnknown *>(this);
00046             ( static_cast<IACFUnknown *>(*ppvObjOut) )->AddRef();
00047             return ACF_OK;
00048         }
00049
00050         return this->CACFUnknown::InternalQueryInterface(riid, ppvObjOut);
00051     }
00052
00053     // CACFUnknown does not support operator=()
00054     AAX_DELETE(AAX_IDataBuffer& operator= (const AAX_IDataBuffer&));
00055 };
00056
00057 #endif
```

15.200 AAX_IDataBufferWrapper.h File Reference

```
#include "AAX.h"
```

Classes

- class [AAX_IDataBufferWrapper](#)
Wrapper for an [AAX_IDataBuffer](#).

Macros

- `#define AAX_IDATABUFFERWRAPPER_H`

15.200.1 Macro Definition Documentation

15.200.1.1 AAX_IDATABUFFERWRAPPER_H

```
#define AAX_IDATABUFFERWRAPPER_H
```

15.201 AAX_IDataBufferWrapper.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019 #ifndef AAX_IDATABUFFERWRAPPER_H
00020 #define AAX_IDATABUFFERWRAPPER_H
00021
00022 #include "AAX.h"
00023
00035 class AAX_IDataBufferWrapper
00036 {
00037 public:
00038     virtual ~AAX_IDataBufferWrapper() = default;
00039
00040     virtual AAX_Result Type(AAX_CTypeID * oType) const = 0;
00041     virtual AAX_Result Size(int32_t * oSize) const = 0;
00042     virtual AAX_Result Data(void const ** oBuffer) const = 0;
00043 };
00044
00045 #endif // AAX_IDATABUFFERWRAPPER_H
```

15.202 AAX_IDescriptionHost.h File Reference

```
#include "AAX.h"
```

Classes

- class [AAX_IDescriptionHost](#)

15.203 AAX_IDescriptionHost.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00011 #ifndef AAXLibrary_AAX_IDescriptionHost_h
00012 #define AAXLibrary_AAX_IDescriptionHost_h
00013
00014 #include "AAX.h"
00015
00016 class AAX_IFeatureInfo;
00017
00018
00021 class AAX_IDescriptionHost
00022 {
00023 public:
00024     virtual ~AAX_IDescriptionHost() {}
00025
00026 public:
00052     virtual const AAX_IFeatureInfo* AcquireFeatureProperties(const AAX_Feature_UID& inFeatureID) const
00053     = 0;
00054 };
00055 #endif
```

15.204 AAX_IDisplayDelegate.h File Reference

```
#include "AAX.h"
```

15.204.1 Description

Defines the display behavior for a parameter.

Classes

- class [AAX_IDisplayDelegateBase](#)
Defines the display behavior for a parameter.
- class [AAX_IDisplayDelegate< T >](#)

15.205 AAX_IDisplayDelegate.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
```

```

00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IDISPLAYDELETGATE_H
00023 #define AAX_IDISPLAYDELETGATE_H
00024
00025 #include "AAX.h"
00026
00027
00028 //Forward declarations
00029 class AAX_CString;
00030
00031 class AAX_IDisplayDelegateBase
00032 {
00033 public:
00034     virtual ~AAX_IDisplayDelegateBase() { }
00035 };
00036
00037 template <typename T>
00038 class AAX_IDisplayDelegate : public AAX_IDisplayDelegateBase
00039 {
00040 public:
00041     virtual AAX_IDisplayDelegate* Clone() const = 0;
00042
00043     virtual bool ValueToString(T value, AAX_CString* valueString) const = 0;
00044
00045     virtual bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const =
00046 0;
00047
00048     virtual bool StringToValue(const AAX_CString& valueString, T* value) const = 0;
00049 };
00050
00051 #endif //AAX_IDISPLAYDELETGATE_H

```

15.206 AAX_IDisplayDelegateDecorator.h File Reference

```
#include "AAX_IDisplayDelegate.h"
```

15.206.1 Description

The base class for all concrete display delegate decorators.

Classes

- class [AAX_IDisplayDelegateDecorator< T >](#)
The base class for all concrete display delegate decorators.

15.207 AAX_IDisplayDelegateDecorator.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003 *
00004 * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005 * All rights reserved.
00006 *
00007 * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008 * not disclose to any third party. Use of the information contained in this
00009 * document is subject to an Avid SDK license.
00010 *

```

```

00011  */
00012
00019  /*=====*/
00020
00021
00022 #ifndef AAX_IDISPLAYDELEGATEDECORATOR_H
00023 #define AAX_IDISPLAYDELEGATEDECORATOR_H
00024
00025 #include "AAX_IDisplayDelegate.h"
00026
00027
00039 template <typename T>
00040 class AAX_IDisplayDelegateDecorator : public AAX_IDisplayDelegate<T>
00041 {
00042 public:
00055     AAX_IDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>& displayDelegate);
00056
00069     AAX_IDisplayDelegateDecorator(const AAX_IDisplayDelegateDecorator& other);
00070
00076     ~AAX_IDisplayDelegateDecorator() AAX_OVERRIDE;
00077
00095     AAX_IDisplayDelegateDecorator<T>* Clone() const AAX_OVERRIDE;
00096
00111     bool ValueToString(T value, AAX_CString* valueString) const AAX_OVERRIDE;
00112
00129     bool ValueToString(T value, int32_t maxNumChars, AAX_CString* valueString) const
AAX_OVERRIDE;
00130
00145     bool StringToValue(const AAX_CString& valueString, T* value) const AAX_OVERRIDE;
00146
00147 private:
00148     const AAX_IDisplayDelegate<T>* mWrappedDisplayDelegate;
00149
00151     AAX_IDisplayDelegateDecorator() { }
00152 };
00153
00154 template <typename T>
00155 AAX_IDisplayDelegateDecorator<T>::AAX_IDisplayDelegateDecorator(const AAX_IDisplayDelegate<T>&
displayDelegate) :
00156     AAX_IDisplayDelegate<T>(),
00157     mWrappedDisplayDelegate(displayDelegate.Clone())
00158 {
00159 }
00160 }
00161
00162 template <typename T>
00163 AAX_IDisplayDelegateDecorator<T>::AAX_IDisplayDelegateDecorator(const AAX_IDisplayDelegateDecorator&
other) :
00164     mWrappedDisplayDelegate(other.mWrappedDisplayDelegate->Clone())
00165 {
00166 }
00167 }
00168
00169 template <typename T>
00170 AAX_IDisplayDelegateDecorator<T>::~AAX_IDisplayDelegateDecorator()
00171 {
00172     delete mWrappedDisplayDelegate;
00173 }
00174
00175 template <typename T>
00176 AAX_IDisplayDelegateDecorator<T>* AAX_IDisplayDelegateDecorator<T>::Clone() const
00177 {
00178     return new AAX_IDisplayDelegateDecorator(*this);
00179 }
00180
00181 template <typename T>
00182 bool AAX_IDisplayDelegateDecorator<T>::ValueToString(T value, AAX_CString* valueString) const
00183 {
00184     return mWrappedDisplayDelegate->ValueToString(value, valueString);
00185 }
00186
00187 template <typename T>
00188 bool AAX_IDisplayDelegateDecorator<T>::ValueToString(T value, int32_t maxNumChars, AAX_CString*
valueString) const
00189 {
00190     return mWrappedDisplayDelegate->ValueToString(value, maxNumChars, valueString);
00191 }
00192
00193 template <typename T>
00194 bool AAX_IDisplayDelegateDecorator<T>::StringToValue(const AAX_CString& valueString, T* value)
const
00195 {
00196     return mWrappedDisplayDelegate->StringToValue(valueString, value);
00197 }
00198
00199
00200

```

```
00201
00202 #endif //AAX_IDISPLAYDELEGATEDECORATOR_H
00203
00204
00205
```

15.208 AAX_IDma.h File Reference

```
#include "AAX.h"
```

15.208.1 Description

Cross-platform interface for access to the host's direct memory access (DMA) facilities.

Classes

- class [AAX_IDma](#)

Cross-platform interface for access to the host's direct memory access (DMA) facilities.

Macros

- #define [AAX_IDMA_H](#)
- #define [AAX_DMA_API](#)

15.208.2 Macro Definition Documentation

15.208.2.1 AAX_IDMA_H

```
#define AAX_IDMA_H
```

15.208.2.2 AAX_DMA_API

```
#define AAX_DMA_API
```

15.209 AAX_IDma.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #pragma once
00023
00024 #ifndef AAX_IDMA_H
00025 #define AAX_IDMA_H
00026
00027 #include "AAX.h"
00028
00029
00030 #ifndef AAX_DMA_API
00031 #   ifdef _MSC_VER
00032 #       define AAX_DMA_API      __cdecl
00033 #   else
00034 #       define AAX_DMA_API
00035 #   endif
00036 #endif // AAX_DMA_API
00037
00049 class AAX_IDma
00050 {
00051 public:
00052     enum EState
00053     {
00054         eState_Error = -1,
00055         eState_Init = 0,
00056         eState_Running = 1,
00057         eState_Complete = 2,
00058         eState_Pending = 3
00059     };
00060
00061     // WARNING! These need to be kept in sync with the TI dMAX microcode EventType IDs!
00062     enum EMode
00063     {
00064         eMode_Error = -1,
00065
00066         eMode_Burst = 6,
00067         eMode_Gather = 10,
00068         eMode_Scatter = 11,
00069     };
00070
00071
00072
00073
00074
00075 };
00076
00077
00078 public:
00079     virtual ~AAX_IDma() {}
00080
00081
00082
00097     virtual AAX_Result      AAX_DMA_API      PostRequest() = 0;
00112     virtual int32_t         AAX_DMA_API      IsTransferComplete() = 0;
00119     virtual AAX_Result      AAX_DMA_API      SetDmaState( EState iState ) = 0;
00122     virtual EState          AAX_DMA_API      GetDmaState() const = 0;
00127     virtual EMode           AAX_DMA_API      GetDmaMode() const = 0;
00129
00130
00132
00145     virtual AAX_Result      AAX_DMA_API      SetSrc( int8_t * iSrc ) = 0;
00148     virtual int8_t *        AAX_DMA_API      GetSrc() = 0;
00157     virtual AAX_Result      AAX_DMA_API      SetDst( int8_t * iDst ) = 0;
00160     virtual int8_t *        AAX_DMA_API      GetDst() = 0;
00161
00169     virtual AAX_Result      AAX_DMA_API      SetBurstLength( int32_t iBurstLengthBytes ) = 0;
00172     virtual int32_t         AAX_DMA_API      GetBurstLength() = 0;
00185     virtual AAX_Result      AAX_DMA_API      SetNumBursts( int32_t iNumBursts ) = 0;
00188     virtual int32_t         AAX_DMA_API      GetNumBursts() = 0;
00196     virtual AAX_Result      AAX_DMA_API      SetTransferSize( int32_t iTransferSizeBytes ) = 0;
00199     virtual int32_t         AAX_DMA_API      GetTransferSize() = 0;
00201
00202
00204
00214     virtual AAX_Result      AAX_DMA_API      SetFifoBuffer( int8_t * iFifoBase ) = 0;

```



```

00217     virtual int8_t *          AAX_DMA_API    GetFifoBuffer() = 0;
00222     virtual AAX_Result        AAX_DMA_API    SetLinearBuffer( int8_t * iLinearBase ) = 0;
00225     virtual int8_t *          AAX_DMA_API    GetLinearBuffer() = 0;
00238     virtual AAX_Result        AAX_DMA_API    SetOffsetTable( const int32_t * iOffsetTable ) = 0;
00241     virtual const int32_t *    AAX_DMA_API    GetOffsetTable() = 0;
00248     virtual AAX_Result        AAX_DMA_API    SetNumOffsets( int32_t iNumOffsets ) = 0;
00251     virtual int32_t           AAX_DMA_API    GetNumOffsets() = 0;
00261     virtual AAX_Result        AAX_DMA_API    SetBaseOffset( int32_t iBaseOffsetBytes ) = 0;
00264     virtual int32_t           AAX_DMA_API    GetBaseOffset() = 0;
00272     virtual AAX_Result        AAX_DMA_API    SetFifoSize( int32_t iSizeBytes ) = 0;
00275     virtual int32_t           AAX_DMA_API    GetFifoSize() = 0;
00277 };
00278
00279
00280
00281 #endif // AAX_IDMA_H

```

15.210 AAX_IEffectDescriptor.h File Reference

```

#include "AAX.h"
#include "AAX_Callbacks.h"

```

15.210.1 Description

Description interface for an effect's (plug-in type's) components.

Classes

- class [AAX_IEffectDescriptor](#)
Description interface for an effect's (plug-in type's) components.

15.211 AAX_IEffectDescriptor.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IEFFECTDESCRIPTOR_H
00023 #define AAX_IEFFECTDESCRIPTOR_H
00024
00025 #include "AAX.h"
00026 #include "AAX_Callbacks.h"
00027
00028 class AAX_IComponentDescriptor;
00029 class AAX_IPropertyMap;
00030
00046 class AAX_IEffectDescriptor
00047 {
00048 public:
00049     virtual ~AAX_IEffectDescriptor() {}
00053     virtual AAX_IComponentDescriptor *      NewComponentDescriptor () = 0;

```

```

00064     virtual AAX_Result          AddComponent ( AAX_IComponentDescriptor*
inComponentDescriptor ) = 0;
00075     virtual AAX_Result          AddName ( const char *inPlugInName ) = 0;
00082     virtual AAX_Result          AddCategory ( uint32_t inCategory ) = 0;
00091     virtual AAX_Result          AddCategoryBypassParameter ( uint32_t inCategory,
AAX_CParamID inParamID ) = 0;
00100     virtual AAX_Result          AddProcPtr ( void * inProcPtr, AAX_CProcPtrID
inProcID ) = 0;
00105     virtual AAX_IPropertyMap *  NewPropertyMap () = 0;
00113     virtual AAX_Result          SetPropertyMap ( AAX_IPropertyMap * inProperties ) =
0;
00122     virtual AAX_Result          AddResourceInfo ( AAX_EResourceType inResourceType,
const char * inInfo ) = 0;
00133     virtual AAX_Result          AddMeterDescription( AAX_CTypeID inMeterID, const
char * inMeterName, AAX_IPropertyMap * inProperties ) = 0;
00151     virtual AAX_Result          AddControlMIDINode ( AAX_CTypeID inNodeID,
AAX_EMIDINodeType inNodeType, const char inNodeName[], uint32_t inChannelMask ) = 0;
00152
00153 };
00154
00155 #endif // AAX_IEFFECTDESCRIPTOR_H

```

15.212 AAX_IEffectDirectData.h File Reference

```

#include "AAX_IACFEEffectDirectData.h"
#include "AAX.h"
#include "CACFUnknown.h"

```

15.212.1 Description

Optional interface for direct access to alg memory.

Classes

- class [AAX_IEffectDirectData](#)
The interface for a AAX Plug-in's direct data interface.

15.213 AAX_IEffectDirectData.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IEFFECTDIRECTDATA_H
00023 #define AAX_IEFFECTDIRECTDATA_H
00024
00025 #include "AAX_IACFEEffectDirectData.h"
00026 #include "AAX.h"
00027 #include "CACFUnknown.h"
00028
00029

```

```

00050 class AAX_IEffectDirectData : public AAX_IACFEEffectDirectData_V2,
00051                               public CACFUnknown
00052 {
00053 public:
00054     ACF_DECLARE_STANDARD_UNKNOWN()
00055
00056     ACFMETHOD(InternalQueryInterface)(const acfiID & riid, void **ppvObjOut) override;
00057
00058     // CACFUnknown does not support operator=()
00059     AAX_DELETE(AAX_IEffectDirectData& operator= (const AAX_IEffectDirectData&));
00060 };
00061
00062 #endif //AAX_IEFFECTDIRECTDATA_H

```

15.214 AAX_IEffectGUI.h File Reference

```

#include "AAX_IACFEEffectGUI.h"
#include "AAX.h"
#include "CACFUnknown.h"

```

15.214.1 Description

The interface for a AAX Plug-in's user interface.

Classes

- class [AAX_IEffectGUI](#)
The interface for a AAX Plug-in's user interface.

15.215 AAX_IEffectGUI.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IEFFECTGUI_H
00023 #define AAX_IEFFECTGUI_H
00024
00025 #include "AAX_IACFEEffectGUI.h"
00026 #include "AAX.h"
00027 #include "CACFUnknown.h"
00028
00029
00050 class AAX_IEffectGUI : public AAX_IACFEEffectGUI,
00051                       public CACFUnknown
00052 {
00053 public:
00054     ACF_DECLARE_STANDARD_UNKNOWN()
00055
00056     ACFMETHOD(InternalQueryInterface)(const acfiID & riid, void **ppvObjOut) override;
00057
00058     // CACFUnknown does not support operator=()
00059     AAX_DELETE(AAX_IEffectGUI& operator= (const AAX_IEffectGUI&));
00060 };
00061
00062 #endif //AAX_IEFFECTGUI_H

```

15.216 AAX_IEffectParameters.h File Reference

```
#include "AAX_IACFEffectParameters.h"
#include "AAX.h"
#include "CACFUnknown.h"
```

15.216.1 Description

The interface for an AAX Plug-in's data model.

Classes

- class [AAX_IEffectParameters](#)
The interface for an AAX Plug-in's data model.

15.217 AAX_IEffectParameters.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IEFFECTPARAMETERS_H
00023 #define AAX_IEFFECTPARAMETERS_H
00024
00025 #include "AAX_IACFEffectParameters.h"
00026 #include "AAX.h"
00027 #include "CACFUnknown.h"
00028
00078 class AAX_IEffectParameters : public AAX_IACFEffectParameters_V4
00079                               , public CACFUnknown
00080 {
00081 public:
00082     ACF_DECLARE_STANDARD_UNKNOWN()
00083
00084     ACF_METHOD(InternalQueryInterface)(const acfiID & riid, void **ppvObjOut) override;
00085
00086     // CACFUnknown does not support operator=()
00087     AAX_DELETE(AAX_IEffectParameters& operator= (const AAX_IEffectParameters&));
00088 };
00089
00090 #endif // AAX_IEFFECTPARAMETERS_H
```

15.218 AAX_IFeatureInfo.h File Reference

```
#include "AAX.h"
```

Classes

- class [AAX_IFeatureInfo](#)

15.219 AAX_IFeatureInfo.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00011 #ifndef AAXLibrary_AAX_IFeatureInfo_h
00012 #define AAXLibrary_AAX_IFeatureInfo_h
00013
00014 #include "AAX.h"
00015
00016
00017 class AAX_IPropertyMap;
00018
00019
00020 class AAX_IFeatureInfo
00021 {
00022 public:
00023     virtual ~AAX_IFeatureInfo() {}
00024
00025 public: // AAX_IACFFeatureInfo
00031     virtual AAX_Result SupportLevel(AAX_ESupportLevel& oSupportLevel) const = 0;
00032
00052     virtual const AAX_IPropertyMap* AcquireProperties() const = 0;
00053
00054 public: // AAX_IFeatureInfo
00057     virtual const AAX_Feature_UID& ID() const = 0;
00058 };
00059
00060
00061 #endif
```

15.220 AAX_IHostProcessor.h File Reference

```
#include "AAX_IACFHostProcessor.h"
#include "AAX.h"
#include "CACFUnknown.h"
```

15.220.1 Description

Base class for the host processor interface which is extended by plugin code.

Classes

- class [AAX_IHostProcessor](#)

Base class for the host processor interface.

15.221 AAX_IHostProcessor.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IHOSTPROCESSOR_H
00023 #define AAX_IHOSTPROCESSOR_H
00024
00025 #include "AAX_IACFHostProcessor.h"
00026 #include "AAX.h"
00027 #include "CACFUnknown.h"
00028
00040 class AAX_IHostProcessor : public AAX_IACFHostProcessor_V2,
00041                             public CACFUnknown
00042 {
00043 public:
00044     ACF_DECLARE_STANDARD_UNKNOWN()
00045
00046     ACFMETHOD(InternalQueryInterface)(const acfIID & riid, void **ppvObjOut) override;
00047
00048     // CACFUnknown does not support operator=()
00049     AAX_DELETE(AAX_IHostProcessor& operator= (const AAX_IHostProcessor&));
00050 };
00051
00052 #endif //AAX_IHOSTPROCESSOR_H

```

15.222 AAX_IHostProcessorDelegate.h File Reference

```
#include "AAX.h"
```

15.222.1 Description

Interface allowing plug-in's HostProcessor to interact with the host's side.

Classes

- class [AAX_IHostProcessorDelegate](#)
Versioned interface for host methods specific to offline processing.

15.223 AAX_IHostProcessorDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do

```

```

00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00018  /*=====*/
00019
00020
00021 #ifndef AAX_IHOSTPROCESSORDELEGATE_H
00022 #define AAX_IHOSTPROCESSORDELEGATE_H
00023
00024 #include "AAX.h"
00025
00037 class AAX_IHostProcessorDelegate
00038 {
00039 public:
00040
00041     virtual ~AAX_IHostProcessorDelegate() {}
00042
00066     virtual AAX_Result    GetAudio ( const float * const inAudioIns [], int32_t inAudioInCount,
int64_t inLocation, int32_t * ioNumSamples ) = 0;
00073     virtual int32_t      GetSideChainInputNum () = 0;
00081     virtual AAX_Result    ForceAnalyze () = 0;
00089     virtual AAX_Result    ForceProcess () = 0;
00090 };
00091
00092 #endif // #ifndef _AAX_IPLUGIN_H_

```

15.224 AAX_IHostServices.h File Reference

```
#include "AAX.h"
```

15.224.1 Description

Various host services.

Classes

- class [AAX_IHostServices](#)

Interface to diagnostic and debugging services provided by the AAX host.

15.225 AAX_IHostServices.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003  *
00004  * Copyright 2014–2015, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019  /*=====*/
00020
00021
00022 #ifndef AAX_IHOSTSERVICES_H
00023 #define AAX_IHOSTSERVICES_H
00024
00025 #include "AAX.h"
00026

```

```

00034 class AAX_IHostServices
00035 {
00036 public:
00037
00038     virtual ~AAX_IHostServices() {}
00039
00058     virtual AAX_Result HandleAssertFailure ( const char * iFile, int32_t iLine, const char * iNote, /*
AAX_EAssertFlags */ int32_t iFlags ) const = 0;
00066     virtual AAX_Result Trace ( int32_t iPriority, const char * iMessage ) const = 0;
00084     virtual AAX_Result StackTrace ( int32_t iTracePriority, int32_t iStackTracePriority, const char *
iMessage ) const = 0;
00085 };
00086
00087 #endif // #ifndef AAX_IHOSTSERVICES_H

```

15.226 AAX_IMIDINode.h File Reference

```

#include "AAX.h"
#include "AAX_ITransport.h"

```

15.226.1 Description

Declaration of the base MIDI Node interface.

Author

by Andriy Goshko

Classes

- class [AAX_IMIDINode](#)
Interface for accessing information in a MIDI node.

15.227 AAX_IMIDINode.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2014-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  *
00010  */
00011 /*=====*/
00019 /*=====*/
00021 #pragma once
00022 #ifndef AAX_IMIDINODE_H
00023 #define AAX_IMIDINODE_H
00025
00026 #include "AAX.h"
00027 #include "AAX_ITransport.h"
00028
00036 class AAX_IMIDINode
00037 {
00038 public:
00039     virtual ~AAX_IMIDINode() {}
00040
00045     virtual AAX_CMidiStream*      GetNodeBuffer () = 0;
00046
00062     virtual AAX_Result            PostMIDIPacket (AAX_CMidiPacket *packet) = 0;
00063
00074     virtual AAX_ITransport*      GetTransport () = 0;
00075 };
00076
00077
00079 #endif // AAX_IMIDINODE_H

```


15.228 AAX_Init.h File Reference

```
#include "AAX.h"
#include "acfbasetypes.h"
```

15.228.1 Description

AAX library implementations of required plug-in initialization, registration, and tear-down methods.

Functions

- [AAX_Result AAXRegisterPlugin](#) ([IACFUnknown](#) *pUnkHost, [IACFPluginDefinition](#) **ppPluginDefinition)
The main plug-in registration method.
- [AAX_Result AAXRegisterComponent](#) ([IACFUnknown](#) *pUnkHost, [acfUInt32](#) index, [IACFComponentDefinition](#) **ppComponentDefinition)
Registers a specific component in the DLL.
- [AAX_Result AAXGetClassFactory](#) ([IACFUnknown](#) *pUnkHost, const [acfCLSID](#) &clsid, const [acfIID](#) &iid, void **ppOut)
Gets the factory for a given class ID.
- [AAX_Result AAXCanUnloadNow](#) ([IACFUnknown](#) *pUnkHost)
Determines whether or not the host may unload the DLL.
- [AAX_Result AAXStartup](#) ([IACFUnknown](#) *pUnkHost)
DLL initialization routine.
- [AAX_Result AAXShutdown](#) ([IACFUnknown](#) *pUnkHost)
DLL shutdown routine.
- [AAX_Result AAXGetSDKVersion](#) ([acfUInt64](#) *oSDKVersion)
Returns the DLL's SDK version.

15.228.2 Function Documentation

15.228.2.1 AAXRegisterComponent()

```
AAX\_Result AAXRegisterComponent (
    IACFUnknown * pUnkHost,
    acfUInt32 index,
    IACFComponentDefinition ** ppComponentDefinition )
```

Registers a specific component in the DLL.

The implementation of this method in the AAX library simply sets *ppComponentDefinition to NULL and returns [AAX_SUCCESS](#).

Wrapped by [ACFRegisterComponent\(\)](#)

Referenced by [ACFRegisterComponent\(\)](#).

Here is the caller graph for this function:



15.228.2.2 AAXGetClassFactory()

```
AAX_Result AAXGetClassFactory (
    IACFUnknown * pUnkHost,
    const acfCLSID & clsid,
    const acfIID & iid,
    void ** ppOut )
```

Gets the factory for a given class ID.

This method is required by ACF but is not supported by [AAX](#). Therefore the implementation of this method in the AAX library simply sets *ppOut to NULL and returns [AAX_ERROR_UNIMPLEMENTED](#).

Wrapped by [ACFGetClassFactory\(\)](#)

Referenced by [ACFGetClassFactory\(\)](#).

Here is the caller graph for this function:



15.228.2.3 AAXCanUnloadNow()

```
AAX_Result AAXCanUnloadNow (
    IACFUnknown * pUnkHost )
```

Determines whether or not the host may unload the DLL.

The implementation of this method in the AAX library returns the result of `GetActiveObjectCount()` as an [AAX_Result](#), with zero active objects interpreted as [AAX_SUCCESS](#) (see `CACFUnknown.h`)

Wrapped by [ACFCanUnloadNow\(\)](#)

Referenced by [ACFCanUnloadNow\(\)](#).

Here is the caller graph for this function:



15.228.2.4 AAXStartup()

```
AAX_Result AAXStartup (
    IACFUnknown * pUnkHost )
```

DLL initialization routine.

Called once at init time. The implementation of this method in the AAX library uses `pUnkHost` as an `IACFComponentFactory` to initialize global services (see `acfbaseapi.h`)

Wrapped by [ACFStartup\(\)](#)

Referenced by [ACFStartup\(\)](#).

Here is the caller graph for this function:



15.228.2.5 AAXShutdown()

```
AAX_Result AAXShutdown (
    IACFUnknown * pUnkHost )
```

DLL shutdown routine.

Called once before unloading the DLL. The implementation of this method in the AAX library tears down any globally initialized state and releases any globally retained resources.

Wrapped by [ACFShutdown\(\)](#)

Referenced by [ACFShutdown\(\)](#).

Here is the caller graph for this function:



15.228.2.6 AAXGetSDKVersion()

```
AAX_Result AAXGetSDKVersion (
    acfUInt64 * oSDKVersion )
```

Returns the DLL's SDK version.

The implementation of this method in the AAX library provides a 64-bit value in which the upper 32 bits represent the SDK version and the lower 32 bits represent the revision number of the SDK. See [AAX_Version.h](#)

Wrapped by [ACFGetSDKVersion\(\)](#)

Referenced by [ACFGetSDKVersion\(\)](#).

Here is the caller graph for this function:



15.229 AAX_Init.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00020 /*=====*/
00021
00022
00023 #include "AAX.h"
00024 #include "acfbasetypes.h"
00025
00026 class IACFUnknown;
00027 class IACFPluginDefinition;
00028 class IACFComponentDefinition;
00029
00030
00046 AAX_Result AAXRegisterPlugin(IACFUnknown * pUnkHost, IACFPluginDefinition **ppPluginDefinition);
00047
00056 AAX_Result AAXRegisterComponent (IACFUnknown * pUnkHost, acfUInt32 index, IACFComponentDefinition
**ppComponentDefinition);
00057
00067 AAX_Result AAXGetClassFactory (IACFUnknown * pUnkHost, const acfCLSID& clsid, const acfIID& iid,
void** ppOut);
00068
00078 AAX_Result AAXCanUnloadNow(IACFUnknown* pUnkHost);
00079
00089 AAX_Result AAXStartup(IACFUnknown* pUnkHost);
00090
00100 AAX_Result AAXShutdown(IACFUnknown* pUnkHost);
00101
00111 AAX_Result AAXGetSDKVersion( acfUInt64 *oSDKVersion );

```

15.230 AAX_IPageTable.h File Reference

```

#include "AAX.h"
#include "AAX_IString.h"

```

Classes

- class [AAX_IPageTable](#)

Interface to the host's representation of a plug-in instance's page table.

15.231 AAX_IPageTable.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00011 #ifndef AAXLibrary_AAX_IPageTable_h

```

```

00012 #define AAXLibrary_AAX_IPageTable_h
00013
00014 #include "AAX.h"
00015 #include "AAX_IString.h"
00016
00017 class IACFUnknown;
00018
00019
00024 class AAX_IPageTable
00025 {
00026 public:
00027
00032     virtual ~AAX_IPageTable()    { }
00033
00034     //
00035     // AAX_IACFPageTable
00036     //
00037
00044     virtual AAX_Result Clear() = 0;
00045
00055     virtual AAX_Result Empty(AAX_CBoolean& oEmpty) const = 0;
00056
00063     virtual AAX_Result GetNumPages(int32_t& oNumPages) const = 0;
00064
00072     virtual AAX_Result InsertPage(int32_t iPage) = 0;
00073
00081     virtual AAX_Result RemovePage(int32_t iPage) = 0;
00082
00096     virtual AAX_Result GetNumMappedParameterIDs(int32_t iPage, int32_t& oNumParameterIdentifiers)
const = 0;
00097
00107     virtual AAX_Result ClearMappedParameter(int32_t iPage, int32_t iIndex) = 0;
00108
00120     virtual AAX_Result GetMappedParameterID(int32_t iPage, int32_t iIndex, AAX_IString&
oParameterIdentifier) const = 0;
00121
00139     virtual AAX_Result MapParameterID(AAX_CPageTableParamID iParameterIdentifier, int32_t iPage,
int32_t iIndex) = 0;
00140
00154     virtual AAX_Result GetNumParametersWithNameVariations(int32_t& oNumParameterIdentifiers) const =
0;
00155
00170     virtual AAX_Result GetNameVariationParameterIDAtIndex(int32_t iIndex, AAX_IString&
oParameterIdentifier) const = 0;
00171
00193     virtual AAX_Result GetNumNameVariationsForParameter(AAX_CPageTableParamID iParameterIdentifier,
int32_t& oNumVariations) const = 0;
00194
00221     virtual AAX_Result GetParameterNameVariationAtIndex(AAX_CPageTableParamID iParameterIdentifier,
int32_t iIndex, AAX_IString& oNameVariation, int32_t& oLength) const = 0;
00222
00243     virtual AAX_Result GetParameterNameVariationOfLength(AAX_CPageTableParamID iParameterIdentifier,
int32_t iLength, AAX_IString& oNameVariation) const = 0;
00244
00252     virtual AAX_Result ClearParameterNameVariations() = 0;
00253
00270     virtual AAX_Result ClearNameVariationsForParameter(AAX_CPageTableParamID iParameterIdentifier) =
0;
00271
00293     virtual AAX_Result SetParameterNameVariation(AAX_CPageTableParamID iParameterIdentifier, const
AAX_IString& iNameVariation, int32_t iLength) = 0;
00294 };
00295
00296 #endif

```

15.232 AAX_IParameter.h File Reference

```
#include "AAX.h"
```

15.232.1 Description

The base interface for all normalizable plug-in parameters.

Classes

- class [AAX_IPParameterValue](#)
An abstract interface representing a parameter value of arbitrary type.
- class [AAX_IPParameter](#)
The base interface for all normalizable plug-in parameters.

15.233 AAX_IPParameter.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012  /*=====*/
00019  /*=====*/
00020
00021
00022  #ifndef AAX_IPARAMETER_H
00023  #define AAX_IPARAMETER_H
00024
00025  #include "AAX.h" //for types
00026
00027  //Forward Declarations
00028  class AAX_CString;
00029  class AAX_IAutomationDelegate;
00030  class AAX_ITaperDelegateBase;
00031  class AAX_IDisplayDelegateBase;
00032  class AAX_IString;
00033
00041  class AAX_IPParameterValue
00042  {
00043  public:
00048      virtual ~AAX_IPParameterValue() { }
00049
00055      virtual AAX_IPParameterValue* Clone() const = 0;
00056
00063      virtual AAX_CParamID Identifier() const = 0;
00064
00077      virtual bool GetValueAsBool(bool* value) const = 0;
00078
00087      virtual bool GetValueAsInt32(int32_t* value) const = 0;
00088
00097      virtual bool GetValueAsFloat(float* value) const = 0;
00098
00107      virtual bool GetValueAsDouble(double* value) const = 0;
00108
00117      virtual bool GetValueAsString(AAX_IString* value) const = 0;
00119  };
00120
00136  class AAX_IPParameter
00137  {
00138  public:
00143      virtual ~AAX_IPParameter() { }
00144
00151      virtual AAX_IPParameterValue* CloneValue() const = 0;
00152
00163      virtual AAX_CParamID Identifier() const = 0;
00164
00174      virtual void SetName(const AAX_CString& name) = 0;
00175
00181      virtual const AAX_CString& Name() const = 0;
00182
00192      virtual void AddShortenedName(const AAX_CString& name) = 0;
00193
00199      virtual const AAX_CString& ShortenedName(int32_t iNumCharacters) const = 0;
00200
00204      virtual void ClearShortenedNames() = 0;
00206
00207
00216      virtual bool Automatable() const = 0;

```

```

00217
00223     virtual void         SetAutomationDelegate( AAX_IAutomationDelegate * iAutomationDelegate ) = 0;
00224
00232     virtual void         Touch() = 0;
00233
00240     virtual void         Release() = 0;
00242
00255     virtual void         SetNormalizedValue(double newNormalizedValue) = 0;
00256
00260     virtual double       GetNormalizedValue() const = 0;
00261
00265     virtual void         SetNormalizedDefaultValue(double normalizedDefault) = 0;
00266
00270     virtual double       GetNormalizedDefaultValue() const = 0;
00271
00275     virtual void         SetToDefaultValue() = 0;
00276
00289     virtual void         SetNumberOfSteps(uint32_t numSteps) = 0;
00290
00295     virtual uint32_t     GetNumberOfSteps() const = 0;
00296
00301     virtual uint32_t     GetStepValue() const = 0;
00302
00307     virtual double       GetNormalizedValueFromStep(uint32_t iStep) const = 0;
00308
00313     virtual uint32_t     GetStepValueFromNormalizedValue(double normalizedValue) const = 0;
00314
00319     virtual void         SetStepValue(uint32_t iStep) = 0;
00320
00322
00323
00338     virtual bool         GetValueString(AAX_CString*   valueString) const = 0;
00339
00350     virtual bool         GetValueString(int32_t iMaxNumChars, AAX_CString*   valueString) const = 0;
00351
00362     virtual bool         GetNormalizedValueFromBool(bool value, double *normalizedValue) const = 0;
00363
00374     virtual bool         GetNormalizedValueFromInt32(int32_t value, double *normalizedValue) const = 0;
00375
00386     virtual bool         GetNormalizedValueFromFloat(float value, double *normalizedValue) const = 0;
00387
00398     virtual bool         GetNormalizedValueFromDouble(double value, double *normalizedValue) const = 0;
00399
00410     virtual bool         GetNormalizedValueFromString(const AAX_CString&   valueString, double
*normalizedValue) const = 0;
00411
00423     virtual bool         GetBoolFromNormalizedValue(double normalizedValue, bool* value) const = 0;
00424
00436     virtual bool         GetInt32FromNormalizedValue(double normalizedValue, int32_t* value) const = 0;
00437
00449     virtual bool         GetFloatFromNormalizedValue(double normalizedValue, float* value) const = 0;
00450
00462     virtual bool         GetDoubleFromNormalizedValue(double normalizedValue, double* value) const = 0;
00463
00475     virtual bool         GetStringFromNormalizedValue(double normalizedValue, AAX_CString&
valueString) const = 0;
00476
00490     virtual bool         GetStringFromNormalizedValue(double normalizedValue, int32_t iMaxNumChars,
AAX_CString&   valueString) const = 0;
00491
00500     virtual bool         SetValueFromString(const AAX_CString&   newValueString) = 0;
00502
00515     virtual bool         GetValueAsBool(bool* value) const = 0;
00516
00525     virtual bool         GetValueAsInt32(int32_t* value) const = 0;
00526
00535     virtual bool         GetValueAsFloat(float* value) const = 0;
00536
00545     virtual bool         GetValueAsDouble(double* value) const = 0;
00546
00555     virtual bool         GetValueAsString(AAX_IString* value) const = 0;
00556
00565     virtual bool         SetValueWithBool(bool value) = 0;
00566
00575     virtual bool         SetValueWithInt32(int32_t value) = 0;
00576
00585     virtual bool         SetValueWithFloat(float value) = 0;
00586
00595     virtual bool         SetValueWithDouble(double value) = 0;
00596
00605     virtual bool         SetValueWithString(const AAX_IString& value) = 0;
00607
00608
00616     virtual void         SetType( AAX_EParameterType iControlType ) = 0;
00617
00622     virtual AAX_EParameterType   GetType() const = 0;
00623

```



```

00624
00630     virtual void          SetOrientation( AAX_EParameterOrientation iOrientation ) = 0;
00631
00635     virtual AAX_EParameterOrientation GetOrientation() const = 0;
00636
00645     virtual void SetTaperDelegate ( AAX_ITaperDelegateBase & inTaperDelegate, bool inPreserveValue ) =
0;
00646
00653     virtual void SetDisplayDelegate ( AAX_IDisplayDelegateBase & inDisplayDelegate ) = 0;
00654
00655 public:
00670     virtual void          UpdateNormalizedValue(double newNormalizedValue) = 0;
00672 };
00673 };
00674
00675 #endif //AAX_IPARAMETER_H
00676
00677
00678
00679

```

15.234 AAX_IPointerQueue.h File Reference

```
#include "AAX_IContainer.h"
```

15.234.1 Description

Abstract interface for a basic FIFO queue of pointers-to-objects.

Classes

- class [AAX_IPointerQueue< T >](#)

15.235 AAX_IPointerQueue.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2015, 2018, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  *
00010  */
00011
00018 /*=====*/
00020 #ifndef AAX_IPOINTERQUEUE_H
00021 #define AAX_IPOINTERQUEUE_H
00023
00024 // AAX Includes
00025 #include "AAX_IContainer.h"
00026
00027
00030 template <typename T>
00031 class AAX_IPointerQueue : public AAX_IContainer
00032 {
00033 public:
00034     virtual ~AAX_IPointerQueue() {}
00035
00036 public:
00037     typedef T template_type;
00038     typedef T* value_type;
00039

```

```

00040 public: // AAX_IContainer
00047     virtual void Clear() = 0;
00048
00049 public: // AAX_IPointerQueue
00056     virtual AAX_IContainer::EStatus Push(value_type inElem) = 0;
00063     virtual value_type Pop() = 0;
00070     virtual value_type Peek() const = 0;
00071 };
00072
00073
00075 #endif /* defined(AAX_IPOINTERQUEUE_H) */

```

15.236 AAX_IPrivateDataAccess.h File Reference

```
#include "AAX.h"
```

15.236.1 Description

Interface to data access provided by host to plug-in.

Classes

- class [AAX_IPrivateDataAccess](#)
Interface to data access provided by host to plug-in.

15.237 AAX_IPrivateDataAccess.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IPRIVATEDATAACCESS_H
00023 #define AAX_IPRIVATEDATAACCESS_H
00024
00025 #include "AAX.h"
00026
00027
00042 class AAX_IPrivateDataAccess
00043 {
00044 public:
00045     virtual ~AAX_IPrivateDataAccess() {}
00046
00061     virtual AAX_Result ReadPortDirect( AAX_CFieldIndex inFieldIndex, const uint32_t
inOffset, const uint32_t inSize, void* outBuffer ) = 0;
00062
00076     virtual AAX_Result WritePortDirect( AAX_CFieldIndex inFieldIndex, const uint32_t
inOffset, const uint32_t inSize, const void* inBuffer ) = 0;
00077 };
00078
00079 #endif //AAX_IPRIVATEDATAACCESS_H

```

15.238 AAX_IPropertyMap.h File Reference

```
#include "AAX_Properties.h"
#include "AAX.h"
```

15.238.1 Description

Generic plug-in description property map.

Classes

- class [AAX_IPropertyMap](#)
Generic plug-in description property map.

15.239 AAX_IPropertyMap.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #ifndef AAX_IPROPERTYMAP_H
00023 #define AAX_IPROPERTYMAP_H
00024
00025 #include "AAX_Properties.h"
00026 #include "AAX.h"
00027
00028 class IACFUnknown;
00029
00055 class AAX_IPropertyMap
00056 {
00057 public:
00058     virtual ~AAX_IPropertyMap() {}
00059
00060     //
00061     // AAX_IACFPropertyMap methods
00062     //
00063     //
00064
00076     virtual AAX_CBoolean      GetProperty ( AAX_EProperty inProperty, AAX_CPropertyValue * outValue
00088 ) const = 0;
00088     virtual AAX_CBoolean      GetPointerProperty ( AAX_EProperty inProperty, const void** outValue )
00089     const = 0;
00101     virtual AAX_Result        AddProperty ( AAX_EProperty inProperty, AAX_CPropertyValue inValue )
00102     = 0;
00117     virtual AAX_Result        AddPointerProperty ( AAX_EProperty inProperty, const void* inValue )
00118     = 0;
00118     virtual AAX_Result        AddPointerProperty ( AAX_EProperty inProperty, const char* inValue )
00119     = 0;
00124     virtual AAX_Result        RemoveProperty ( AAX_EProperty inProperty ) = 0;
00125
00136     virtual AAX_Result        AddPropertyWithIDArray ( AAX_EProperty inProperty, const
00137 AAX_SPlugInIdentifierTriad* inPluginIDs, uint32_t inNumPluginIDs) = 0;
00148     virtual AAX_CBoolean      GetPropertyWithIDArray ( AAX_EProperty inProperty, const
00149 AAX_SPlugInIdentifierTriad* outPluginIDs, uint32_t* outNumPluginIDs) const = 0;
```

```

00149
00150
00151     //
00152     // AAX_IPropertyMap methods
00153     //
00154
00157     virtual IACFUnknown*      GetIUnknown() = 0;
00158 };
00159
00160 #endif // AAX_IPROPERTYMAP_H

```

15.240 AAX_ISessionDocument.h File Reference

```

#include "AAX_SessionDocumentTypes.h"
#include "AAX_UIDs.h"
#include "AAX.h"
#include <memory>

```

Classes

- class [AAX_ISessionDocument](#)
Interface representing information in a host session document.
- class [AAX_ISessionDocument::TempoMap](#)

Macros

- #define [AAX_ISessionDocument_H](#)

15.240.1 Macro Definition Documentation

15.240.1.1 AAX_ISessionDocument_H

```
#define AAX_ISessionDocument_H
```

15.241 AAX_ISessionDocument.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017

```

```

00018 #pragma once
00019 #ifndef AAX_I_SessionDocument_H
00020 #define AAX_I_SessionDocument_H
00021
00022 #include "AAX_SessionDocumentTypes.h"
00023 #include "AAX_UIDs.h"
00024 #include "AAX.h"
00025 #include <memory>
00026
00027 class IACFUnknown;
00028
00029 class AAX_I_SessionDocument
00030 {
00031 public:
00032     virtual ~AAX_I_SessionDocument() = default;
00033
00034     class TempoMap
00035     {
00036     public:
00037         virtual ~TempoMap() = default;
00038         virtual int32_t Size() const = 0;
00039         virtual AAX_CTempoBreakpoint const * Data() const = 0;
00040     };
00041
00042     virtual bool Valid() const = 0;
00043
00044     virtual std::unique_ptr<TempoMap const> GetTempoMap() = 0;
00045
00046     virtual AAX_Result GetDocumentData(AAX_DocumentData_UID const & inDataType, IACFUnknown **
        outData) = 0;
00047 };
00048
00049 #endif // AAX_I_SessionDocument_H

```

15.242 AAX_I_SessionDocumentClient.h File Reference

```

#include "AAX_IACFSessionDocumentClient.h"
#include "AAX.h"
#include "CACFUnknown.h"

```

Classes

- class [AAX_I_SessionDocumentClient](#)
Interface representing a client of the session document interface.

Macros

- #define [AAX_I_SessionDocumentClient_H](#)

15.242.1 Macro Definition Documentation

15.242.1.1 AAX_I_SessionDocumentClient_H

```
#define AAX_I_SessionDocumentClient_H
```

15.243 AAX_I_SessionDocumentClient.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019 #ifndef AAX_I_SessionDocumentClient_H
00020 #define AAX_I_SessionDocumentClient_H
00021
00022 #include "AAX_IACFSessionDocumentClient.h"
00023 #include "AAX.h"
00024 #include "CACFUnknown.h"
00025
00026
00033 class AAX_I_SessionDocumentClient : public AAX_IACFSessionDocumentClient,
00034                                     public CACFUnknown
00035 {
00036 public:
00037     ACF_DECLARE_STANDARD_UNKNOWN()
00038
00039     ACFMETHOD(InternalQueryInterface)(const acfIID & riid, void **ppvObjOut) override;
00040
00041     // CACFUnknown does not support operator=()
00042     AAX_DELETE(AAX_I_SessionDocumentClient& operator= (const AAX_I_SessionDocumentClient&));
00043 };
00044
00045 #endif //AAX_I_SessionDocumentClient_H

```

15.244 AAX_IString.h File Reference

```
#include "AAX.h"
```

15.244.1 Description

An AAX string interface.

Classes

- class [AAX_IString](#)

A simple string container that can be passed across a binary boundary. This class, for simplicity, is not versioned and thus can never change.

15.245 AAX_IString.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00019  /*=====*/
00020
00021
00022  #ifndef AAX_ISTRING_H
00023  #define AAX_ISTRING_H
00024
00025  #include "AAX.h"    //for types
00026
00027
00037  class AAX_IString
00038  {
00039  public:
00041      virtual ~AAX_IString () {}
00042
00044      virtual uint32_t      Length () const = 0;
00045      virtual uint32_t      MaxLength () const = 0;
00046
00048      virtual const char *  Get () const = 0;
00049      virtual void          Set ( const char * iString ) = 0;
00050
00052      virtual AAX_IString & operator=(const AAX_IString & iOther) = 0;
00053      virtual AAX_IString & operator=(const char * iString) = 0;
00054  };
00055
00056
00057
00058
00059  #endif //AAX_ISTRING_H

```

15.246 AAX_ITaperDelegate.h File Reference

15.246.1 Description

Defines the taper conversion behavior for a parameter.

Classes

- class [AAX_ITaperDelegateBase](#)
Defines the taper conversion behavior for a parameter.
- class [AAX_ITaperDelegate< T >](#)

15.247 AAX_ITaperDelegate.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2014-2015, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *

```

```

00007  *   CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  *   not disclose to any third party. Use of the information contained in this
00009  *   document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019  /*=====*/
00020
00021
00022 #ifndef AAX_ITAPERDELEGATE_H
00023 #define AAX_ITAPERDELEGATE_H
00024
00025
00026
00069 class AAX_ITaperDelegateBase
00070 {
00071 public:
00076     virtual ~AAX_ITaperDelegateBase()          { }
00077 };
00078
00084 template <typename T>
00085 class AAX_ITaperDelegate : public AAX_ITaperDelegateBase
00086 {
00087 public:
00101     virtual AAX_ITaperDelegate*   Clone() const = 0;
00102
00106     virtual T                     GetMaximumValue() const = 0;
00107
00111     virtual T                     GetMinimumValue() const = 0;
00112
00124     virtual T                     ConstrainRealValue(T value) const = 0;
00125
00136     virtual T                     NormalizedToReal(double normalizedValue) const = 0;
00137
00148     virtual double                RealToNormalized(T realValue) const = 0;
00149 };
00150
00151
00152
00153 #endif //AAX_ITAPERDELEGATE_H

```

15.248 AAX_ITask.h File Reference

```

#include "AAX_IACFTask.h"
#include "AAX.h"

```

Classes

- class [AAX_ITask](#)
Interface representing a request to perform a task.

Macros

- #define [AAX_ITask_H](#)

15.248.1 Macro Definition Documentation

15.248.1.1 AAX_ITask_H

```
#define AAX_ITask_H
```


15.249 AAX_ITask.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00016  /*=====*/
00017
00018  #pragma once
00019
00020  #ifndef AAX_ITask_H
00021  #define AAX_ITask_H
00022
00023  #include "AAX_IACFTask.h"
00024  #include "AAX.h"
00025
00026  class AAX_IACFDataBuffer;
00027
00028
00047  class AAX_ITask
00048  {
00049  public:
00050      virtual ~AAX_ITask() = default;
00051
00058      virtual AAX_Result GetType(AAX_CTypeID * oType) const = 0;
00059
00073      virtual AAX_IACFDataBuffer const * GetArgumentOfType(AAX_CTypeID iType) const = 0;
00074
00081      virtual AAX_Result SetProgress(float iProgress) = 0;
00082
00086      virtual float GetProgress() const = 0;
00087
00106      virtual AAX_Result AddResult(AAX_IACFDataBuffer const * iResult) = 0;
00107
00125      virtual AAX_ITask * SetDone(AAX_TaskCompletionStatus iStatus) = 0;
00126  };
00127
00128
00129  #endif

```

15.250 AAX_ITaskAgent.h File Reference

```

#include "AAX_IACFTaskAgent.h"
#include "AAX.h"
#include "CACFUnknown.h"

```

Classes

- class [AAX_ITaskAgent](#)
Interface for a component that accepts task requests.

15.251 AAX_ITaskAgent.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*

```

```

00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016  /*=====*/
00017
00018
00019 #ifndef AAX_ITaskAgent_H
00020 #define AAX_ITaskAgent_H
00021
00022 #include "AAX_IACFTaskAgent.h"
00023 #include "AAX.h"
00024 #include "CACFUnknown.h"
00025
00026
00034 class AAX_ITaskAgent : public AAX_IACFTaskAgent
00035                       , public CACFUnknown
00036 {
00037 public:
00038     ACF_DECLARE_STANDARD_UNKNOWN()
00039
00040     ACFMETHOD(InternalQueryInterface)(const acfIID & riid, void **ppvObjOut) AAX_OVERRIDE;
00041
00042     // CACFUnknown does not support operator=()
00043     AAX_DELETE(AAX_ITaskAgent& operator= (const AAX_ITaskAgent&));
00044 };
00045
00046 #endif //AAX_IEFFECTDIRECTDATA_H

```

15.252 AAX_ITransport.h File Reference

```

#include "AAX.h"
#include "AAX_Enums.h"

```

15.252.1 Description

The interface for query ProTools transport information.

Note

To use this interface plug-in must describe AAX_eProperty_UsesMIDI property

Classes

- class [AAX_ITransport](#)

Interface to information about the host's transport state.

15.253 AAX_ITransport.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2018-2021, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012  /*=====*/
00020  /*=====*/
00021
00022
00023 #ifndef AAX_ITRANSPORT_H
00024 #define AAX_ITRANSPORT_H
00025
00026 #pragma once
00027
00028 #include "AAX.h"
00029 #include "AAX_Enums.h"
00030
00052 class AAX_ITransport
00053 {
00054 public:
00055
00060     virtual ~AAX_ITransport() {}
00061
00072     virtual AAX_Result GetCurrentTempo ( double* TempoBPM ) const = 0;
00073
00082     virtual AAX_Result GetCurrentMeter ( int32_t* MeterNumerator, int32_t* MeterDenominator ) const
00083     = 0;
00089     virtual AAX_Result IsTransportPlaying ( bool* isPlaying ) const = 0;
00090
00103     virtual AAX_Result GetCurrentTickPosition ( int64_t* TickPosition ) const = 0;
00104
00116     virtual AAX_Result GetCurrentLoopPosition ( bool* bLooping, int64_t* LoopStartTick, int64_t*
00117     LoopEndTick ) const = 0;
00136     virtual AAX_Result GetCurrentNativeSampleLocation ( int64_t* SampleLocation ) const = 0;
00137
00147     virtual AAX_Result GetCustomTickPosition( int64_t* oTickPosition, int64_t iSampleLocation)
00148     const = 0;
00160     virtual AAX_Result GetBarBeatPosition(int32_t* Bars, int32_t* Beats, int64_t* DisplayTicks,
00161     int64_t SampleLocation) const = 0;
00167     virtual AAX_Result GetTicksPerQuarter ( uint32_t* ticks ) const = 0;
00168
00174     virtual AAX_Result GetCurrentTicksPerBeat ( uint32_t* ticks ) const = 0;
00175
00183     virtual AAX_Result GetTimelineSelectionStartPosition ( int64_t* oSampleLocation ) const = 0;
00184
00193     virtual AAX_Result GetTimeCodeInfo( AAX_EFrameRate* oFrameRate, int32_t* oOffset ) const = 0;
00194
00203     virtual AAX_Result GetFeetFramesInfo( AAX_EFeetFramesRate* oFeetFramesRate, int64_t* oOffset )
00204     const = 0;
00212     virtual AAX_Result IsMetronomeEnabled ( int32_t* isEnabled ) const = 0;
00213
00222     virtual AAX_Result GetHDDTimeCodeInfo( AAX_EFrameRate* oHDDFrameRate, int64_t* oHDDOffset ) const
00223     = 0;
00229     virtual AAX_Result RequestTransportStart() = 0;
00230
00236     virtual AAX_Result RequestTransportStop() = 0;
00237
00245     virtual AAX_Result GetTimelineSelectionEndPosition ( int64_t* oSampleLocation ) const = 0;
00246 };
00247
00248 #endif // AAX_ITRANSPORT_H
00249

```

15.254 AAX_IViewContainer.h File Reference

```

#include "AAX_GUITypes.h"
#include "AAX.h"

```

15.254.1 Description

Interface for the AAX host's view of a single instance of an effect.

Classes

- class [AAX_IViewContainer](#)

Interface for the AAX host's view of a single instance of an effect. Used both by clients of the AAX host and by effect components.

15.255 AAX_IViewContainer.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2017, 2019, 2021, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012  /*=====*/
00019  /*=====*/
00020
00021
00022  #ifndef _AAX_IVIEWCONTAINER_H_
00023  #define _AAX_IVIEWCONTAINER_H_
00024
00025  #include "AAX_GUITypes.h"
00026  #include "AAX.h"
00027
00028
00037  class AAX_IViewContainer
00038  {
00039  public:
00040      virtual ~AAX_IViewContainer(void) {}
00041
00047      virtual int32_t      GetType () = 0;
00050      virtual void *      GetPtr () = 0;
00062      virtual AAX_Result   GetModifiers ( uint32_t * outModifiers ) = 0;
00064
00077      virtual AAX_Result   SetViewSize ( AAX_Point & inSize ) = 0;
00079
00111      virtual AAX_Result   HandleParameterMouseDown ( AAX_CParamID inParamID, uint32_t inModifiers ) =
00124      0;
00124      virtual AAX_Result   HandleParameterMouseDrag ( AAX_CParamID inParamID, uint32_t inModifiers ) =
00137      0;
00137      virtual AAX_Result   HandleParameterMouseUp ( AAX_CParamID inParamID, uint32_t inModifiers ) = 0;
00138
00148      virtual AAX_Result   HandleParameterMouseEnter ( AAX_CParamID inParamID, uint32_t inModifiers ) =
00149      0;
00149      virtual AAX_Result   HandleParameterMouseExit ( AAX_CParamID inParamID, uint32_t inModifiers ) =
00160      0;
00170      virtual AAX_Result   HandleMultipleParametersMouseDown ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) = 0;
00185      virtual AAX_Result   HandleMultipleParametersMouseDrag ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) = 0;
00200      virtual AAX_Result   HandleMultipleParametersMouseUp ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) = 0;
00202  };
00203
00204  #endif
00205

```

15.256 AAX_MIDIUtilities.h File Reference

```
#include "AAX.h"
```

15.256.1 Description

Utilities for managing MIDI data.

Namespaces

- namespace [AAX](#)

Enumerations

- enum [AAX::EStatusNibble](#) {
[AAX::eStatusNibble_NoteOff](#) = 0x80 ,
[AAX::eStatusNibble_NoteOn](#) = 0x90 ,
[AAX::eStatusNibble_KeyPressure](#) = 0xA0 ,
[AAX::eStatusNibble_ControlChange](#) = 0xB0 ,
[AAX::eStatusNibble_ChannelMode](#) = 0xB0 ,
[AAX::eStatusNibble_ProgramChange](#) = 0xC0 ,
[AAX::eStatusNibble_ChannelPressure](#) = 0xD0 ,
[AAX::eStatusNibble_PitchBend](#) = 0xE0 ,
[AAX::eStatusNibble_SystemCommon](#) = 0xF0 ,
[AAX::eStatusNibble_SystemRealTime](#) = 0xF0 }
Values for the status nibble in a MIDI packet.
- enum [AAX::EStatusByte](#) {
[AAX::eStatusByte_SysExBegin](#) = 0xF0 ,
[AAX::eStatusByte_MTCQuarterFrame](#) = 0xF1 ,
[AAX::eStatusByte_SongPosition](#) = 0xF2 ,
[AAX::eStatusByte_SongSelect](#) = 0xF3 ,
[AAX::eStatusByte_TuneRequest](#) = 0xF6 ,
[AAX::eStatusByte_SysExEnd](#) = 0xF7 ,
[AAX::eStatusByte_TimingClock](#) = 0xF8 ,
[AAX::eStatusByte_Start](#) = 0xFA ,
[AAX::eStatusByte_Continue](#) = 0xFB ,
[AAX::eStatusByte_Stop](#) = 0xFC ,
[AAX::eStatusByte_ActiveSensing](#) = 0xFE ,
[AAX::eStatusByte_Reset](#) = 0xFF }
Values for the status byte in a MIDI packet.
- enum [AAX::EChannelModeData](#) {
[AAX::eChannelModeData_AllSoundOff](#) = 120 ,
[AAX::eChannelModeData_ResetControllers](#) = 121 ,
[AAX::eChannelModeData_LocalControl](#) = 122 ,
[AAX::eChannelModeData_AllNotesOff](#) = 123 ,
[AAX::eChannelModeData_OmniOff](#) = 124 ,
[AAX::eChannelModeData_OmniOn](#) = 125 ,
[AAX::eChannelModeData_PolyOff](#) = 126 ,
[AAX::eChannelModeData_PolyOn](#) = 127 }
Values for the first data byte in a Channel Mode Message MIDI packet.
- enum [AAX::ESpecialData](#) {
[AAX::eSpecialData_AccentedClick](#) = 0x00 ,
[AAX::eSpecialData_UnaccentedClick](#) = 0x01 }
Special message data for the first data byte in a message.

Functions

- bool `AAX::IsNoteOn` (const `AAX_CMidiPacket` *inPacket)
Returns true if inPacket is a Note On message.
- bool `AAX::IsNoteOff` (const `AAX_CMidiPacket` *inPacket)
Returns true if inPacket is a Note Off message, or a Note On message with velocity zero.
- bool `AAX::IsAllNotesOff` (const `AAX_CMidiPacket` *inPacket)
Returns true if inPacket is an All Sound Off or All Notes Off message.
- bool `AAX::IsAccentedClick` (const `AAX_CMidiPacket` *inPacket)
Returns true if inPacket is a special Pro Tools accented click message.
- bool `AAX::IsUnaccentedClick` (const `AAX_CMidiPacket` *inPacket)
Returns true if inPacket is a special Pro Tools unaccented click message.
- bool `AAX::IsClick` (const `AAX_CMidiPacket` *inPacket)
Returns true if inPacket is a special Pro Tools click message.

15.257 AAX_MIDIUtilities.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00019 #ifndef AAX_MIDIUtilities_h
00020 #define AAX_MIDIUtilities_h
00022
00023 // AAX Includes
00024 #include "AAX.h"
00025
00026 namespace AAX
00027 {
00028     //
00029     // Some MIDI defines
00030     //
00031
00032     enum EStatusNibble
00033     {
00034         eStatusNibble_NoteOff = 0x80
00035         , eStatusNibble_NoteOn = 0x90
00036         , eStatusNibble_KeyPressure = 0xA0
00037         , eStatusNibble_ControlChange = 0xB0
00038         , eStatusNibble_ChannelMode = 0xB0
00039         , eStatusNibble_ProgramChange = 0xC0
00040         , eStatusNibble_ChannelPressure = 0xD0
00041         , eStatusNibble_PitchBend = 0xE0
00042         , eStatusNibble_SystemCommon = 0xF0
00043         , eStatusNibble_SystemRealTime = 0xF0
00044     };
00045
00046     enum EStatusByte
00047     {
00048         eStatusByte_SysExBegin = 0xF0
00049         , eStatusByte_MTCQuarterFrame = 0xF1
00050         , eStatusByte_SongPosition = 0xF2
00051         , eStatusByte_SongSelect = 0xF3
00052         , eStatusByte_TuneRequest = 0xF6
00053         , eStatusByte_SysExEnd = 0xF7
00054         , eStatusByte_TimingClock = 0xF8
00055         , eStatusByte_Start = 0xFA
00056         , eStatusByte_Continue = 0xFB
00057         , eStatusByte_Stop = 0xFC
00058         , eStatusByte_ActiveSensing = 0xFE
00059         , eStatusByte_Reset = 0xFF
00060     };
00061
00062     enum EChannelModeData

```

```

00066     {
00067         eChannelModeData_AllSoundOff = 120
00068         , eChannelModeData_ResetControllers = 121
00069         , eChannelModeData_LocalControl = 122
00070         , eChannelModeData_AllNotesOff = 123
00071         , eChannelModeData_OmniOff = 124
00072         , eChannelModeData_OmniOn = 125
00073         , eChannelModeData_PolyOff = 126
00074         , eChannelModeData_PolyOn = 127
00075     };
00076
00077     enum ESpecialData
00078     {
00079         eSpecialData_AccentedClick = 0x00
00080         , eSpecialData_UnaccentedClick = 0x01
00081     };
00082 };
00083
00084
00085 //
00086 // Basic MIDI utility functions
00087 //
00088
00089 inline bool IsNoteOn(const AAX_CMidiPacket* inPacket)
00090 {
00091     if (!inPacket) { return false; }
00092     const uint8_t sn = (inPacket->mData[0] & 0xF0); // status nibble
00093     const uint8_t data2 = inPacket->mData[2];
00094     return ((eStatusNibble_NoteOn == sn) &&
00095         (0x00 != data2));
00096 }
00097
00098
00099 inline bool IsNoteOff(const AAX_CMidiPacket* inPacket)
00100 {
00101     if (!inPacket) { return false; }
00102     const uint8_t sn = (inPacket->mData[0] & 0xF0); // status nibble
00103     const uint8_t data2 = inPacket->mData[2];
00104     return ((eStatusNibble_NoteOff == sn) || ((eStatusNibble_NoteOn == sn) && (0x00 == data2)));
00105 }
00106
00107
00108 inline bool IsAllNotesOff(const AAX_CMidiPacket* inPacket)
00109 {
00110     if (!inPacket) { return false; }
00111     const uint8_t sn = (inPacket->mData[0] & 0xF0); // status nibble
00112     const uint8_t data1 = inPacket->mData[1];
00113     const uint8_t data2 = inPacket->mData[2];
00114     if (eStatusNibble_ChannelMode == sn)
00115     {
00116         if (eChannelModeData_PolyOff == data1)
00117         {
00118             return true;
00119         }
00120         else if ((eChannelModeData_AllSoundOff == data1) ||
00121             (eChannelModeData_AllNotesOff == data1) ||
00122             (eChannelModeData_OmniOff == data1) ||
00123             (eChannelModeData_OmniOn == data1) ||
00124             (eChannelModeData_PolyOn == data1))
00125         {
00126             return (0x00 == data2);
00127         }
00128     }
00129     return false;
00130 }
00131
00132
00133 inline bool IsAccentedClick(const AAX_CMidiPacket* inPacket)
00134 {
00135     return ((inPacket) &&
00136         (eStatusNibble_NoteOn == (inPacket->mData[0] & 0xF0)) &&
00137         (0x00 == (inPacket->mData[0] & 0x0F)) &&
00138         (eSpecialData_AccentedClick == inPacket->mData[1]));
00139 }
00140
00141
00142 inline bool IsUnaccentedClick(const AAX_CMidiPacket* inPacket)
00143 {
00144     return ((inPacket) &&
00145         (eStatusNibble_NoteOn == (inPacket->mData[0] & 0xF0)) &&
00146         (0x00 == (inPacket->mData[0] & 0x0F)) &&
00147         (eSpecialData_UnaccentedClick == inPacket->mData[1]));
00148 }
00149
00150
00151 inline bool IsClick(const AAX_CMidiPacket* inPacket)
00152 {
00153     return (IsAccentedClick(inPacket) || IsUnaccentedClick(inPacket));
00154 }
00155 } // namespace AAX
00156
00157 #endif // AAX_MIDIUtilities_h

```

15.258 AAX_PageTableUtilities.h File Reference

```
#include "AAX_CString.h"
#include "AAX.h"
```

Namespaces

- namespace [AAX](#)

Functions

- `template<class T1 , class T2 >`
`bool AAX::PageTableParameterMappingsAreEqual (const T1 &inL, const T2 &inR)`
- `template<class T1 , class T2 >`
`bool AAX::PageTableParameterNameVariationsAreEqual (const T1 &inL, const T2 &inR)`
- `template<class T1 , class T2 >`
`bool AAX::PageTablesAreEqual (const T1 &inL, const T2 &inR)`
- `template<class T >`
`void AAX::CopyPageTable (T &to, const T &from)`
- `template<class T >`
`std::vector< std::pair< int32_t, int32_t > > AAX::FindParameterMappingsInPageTable (const T &inTable, AAX_CParamID inParameterID)`
- `template<class T >`
`void AAX::ClearMappedParameterByID (T &ioTable, AAX_CParamID inParameterID)`

15.259 AAX_PageTableUtilities.h

[Go to the documentation of this file.](#)

```
00001  /*=====*/
00002  /*
00003   * Copyright 2016-2017, 2023 Avid Technology, Inc.
00004   * All rights reserved.
00005   *
00006   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007   * not disclose to any third party. Use of the information contained in this
00008   * document is subject to an Avid SDK license.
00009   */
00010
00011 #ifndef AAXLibrary_AAX_PageTableUtilities_h
00012 #define AAXLibrary_AAX_PageTableUtilities_h
00013
00014 #include "AAX_CString.h"
00015 #include "AAX.h"
00016
00017 namespace AAX
00018 {
00019     template <class T1, class T2>
00020     inline bool PageTableParameterMappingsAreEqual(const T1& inL, const T2& inR)
00021     {
00022         AAX_Result errL = AAX_SUCCESS;
00023         AAX_Result errR = AAX_SUCCESS;
00024
00025         int32_t numPagesL = -1;
00026         int32_t numPagesR = -1;
00027         errL = inL.GetNumPages(numPagesL);
00028         errR = inR.GetNumPages(numPagesR);
00029
00030         if (errL != errR || numPagesL != numPagesR) { return false; }
00031         else if (AAX_SUCCESS != errL) { return true; } // can't get page data from either table
00032
00033         for (int32_t i = 0; i < numPagesL; ++i)
00034         {
```



```

00039         int32_t numParamsL = -1;
00040         int32_t numParamsR = -1;
00041         errL = inL.GetNumMappedParameterIDs(i, numParamsL);
00042         errR = inR.GetNumMappedParameterIDs(i, numParamsR);
00043
00044         if (errL != errR || numParamsL != numParamsR) { return false; }
00045         else if (AAX_SUCCESS != errL) { continue; } // skip this page if equal errors were
returned
00046
00047         for (int32_t j = 0; j < numParamsL; ++j)
00048         {
00049             AAX_CString paramIdentifierL;
00050             AAX_CString paramIdentifierR;
00051             errL = inL.GetMappedParameterID(i, j, paramIdentifierL);
00052             errR = inR.GetMappedParameterID(i, j, paramIdentifierR);
00053
00054             if (errL != errR || paramIdentifierL != paramIdentifierR) { return false; }
00055         }
00056     }
00057
00058     return true;
00059 }
00060
00061 template <class T1, class T2>
00062 inline bool PageTableParameterNameVariationsAreEqual(const T1& inL, const T2& inR)
00063 {
00064     AAX_Result errL = AAX_SUCCESS;
00065     AAX_Result errR = AAX_SUCCESS;
00066
00067     int32_t numParamIdentifiersL = -1;
00068     int32_t numParamIdentifiersR = -1;
00069     errL = inL.GetNumParametersWithNameVariations(numParamIdentifiersL);
00070     errR = inR.GetNumParametersWithNameVariations(numParamIdentifiersR);
00071
00072     if (errL != errR || numParamIdentifiersL != numParamIdentifiersR) { return false; }
00073     else if (AAX_SUCCESS != errL) { return true; } // can't get parameter name variation data from
either table
00074
00075     for (int32_t i = 0; i < numParamIdentifiersL; ++i)
00076     {
00077         AAX_CString paramIdentifierL;
00078         AAX_CString paramIdentifierR;
00079         errL = inL.GetNameVariationParameterIDAtIndex(i, paramIdentifierL);
00080         errR = inR.GetNameVariationParameterIDAtIndex(i, paramIdentifierR);
00081
00082         if (errL != errR || paramIdentifierL != paramIdentifierR) { return false; }
00083         else if (AAX_SUCCESS != errL) { continue; } // skip this index if equal errors were
returned
00084
00085         int32_t numVariationsL = -1;
00086         int32_t numVariationsR = -1;
00087         errL = inL.GetNumNameVariationsForParameter(paramIdentifierL.Get(), numVariationsL);
00088         errR = inR.GetNumNameVariationsForParameter(paramIdentifierR.Get(), numVariationsR);
00089
00090         if (errL != errR || numVariationsL != numVariationsR) { return false; }
00091         else if (AAX_SUCCESS != errL) { continue; } // skip this index if equal errors were
returned
00092
00093         for (int32_t j = 0; j < numVariationsL; ++j)
00094         {
00095             AAX_CString nameVariationL;
00096             int32_t lengthL;
00097             AAX_CString nameVariationR;
00098             int32_t lengthR;
00099             errL = inL.GetParameterNameVariationAtIndex(paramIdentifierL.Get(), j, nameVariationL,
lengthL);
00100             errR = inR.GetParameterNameVariationAtIndex(paramIdentifierR.Get(), j, nameVariationR,
lengthR);
00101
00102             if (errL != errR || lengthL != lengthR || nameVariationL != nameVariationR) { return
false; }
00103         }
00104     }
00105
00106     return true;
00107 }
00108
00109 template <class T1, class T2>
00110 inline bool PageTablesAreEqual(const T1& inL, const T2& inR)
00111 {
00112     return (PageTableParameterMappingsAreEqual(inL, inR) &&
PageTableParameterNameVariationsAreEqual(inL, inR));
00113 }
00114
00115 template <class T>
00116 inline void CopyPageTable(T& to, const T& from)
00117 {

```

```

00122         to.Clear();
00123
00124         // Copy page tables
00125         int32_t curPageIndex;
00126         from.GetNumPages(curPageIndex);
00127         while (0 < curPageIndex--)
00128         {
00129             to.InsertPage(0);
00130
00131             int32_t numIDsRemaining = 0;
00132             from.GetNumMappedParameterIDs(curPageIndex, numIDsRemaining);
00133             for (int32_t curSlotIndex = 0; 0 < numIDsRemaining; ++curSlotIndex) // numIDsRemaining is
decremented in the loop body
00134             {
00135                 AAX_CString curParam;
00136                 const AAX_Result getResult = from.GetMappedParameterID(curPageIndex,
curSlotIndex, curParam);
00137                 if (AAX_SUCCESS == getResult)
00138                 {
00139                     to.MapParameterID(curParam.CString(), 0, curSlotIndex);
00140                     --numIDsRemaining;
00141                 }
00142             }
00143         }
00144
00145         // Copy name variations
00146         int32_t numParameterIdentifiers = 0;
00147         to.ClearParameterNameVariations();
00148         from.GetNumParametersWithNameVariations(numParameterIdentifiers);
00149         for (int32_t curParamIndex = 0; curParamIndex < numParameterIdentifiers; ++curParamIndex)
00150         {
00151             AAX_CString curParamIdentifier;
00152             from.GetNameVariationParameterIDAtIndex(curParamIndex, curParamIdentifier);
00153
00154             int32_t numNameVariations = 0;
00155             from.GetNumNameVariationsForParameter(curParamIdentifier.Get(), numNameVariations);
00156
00157             for (int32_t curNameVariationIndex = 0; curNameVariationIndex < numNameVariations;
++curNameVariationIndex)
00158             {
00159                 int32_t curNameVariationLength;
00160                 AAX_CString curNameVariation;
00161                 from.GetParameterNameVariationAtIndex(curParamIdentifier.Get(), curNameVariationIndex,
curNameVariation, curNameVariationLength);
00162                 to.SetParameterNameVariation(curParamIdentifier.Get(), curNameVariation,
curNameVariationLength);
00163             }
00164         }
00165     }
00166
00167     template <class T>
00173     inline std::vector<std::pair<int32_t, int32_t> > FindParameterMappingsInPageTable(const T&
inTable, AAX_CParamID inParameterID)
00174     {
00175         const AAX_CString searchParamID(inParameterID);
00176         std::vector<std::pair<int32_t, int32_t> > foundParamMappings;
00177
00178         int32_t numPages = 0;
00179         inTable.GetNumPages(numPages);
00180         for (int32_t i = 0; i < numPages; ++i)
00181         {
00182             int32_t numIDsRemaining = 0;
00183             inTable.GetNumMappedParameterIDs(i, numIDsRemaining);
00184             for (int32_t curSlotIndex = 0; 0 < numIDsRemaining; ++curSlotIndex) // numIDs is
decremented in the loop body
00185             {
00186                 AAX_CString curParam;
00187                 const AAX_Result getResult = inTable.GetMappedParameterID(i, curSlotIndex,
curParam);
00188                 if (AAX_SUCCESS == getResult)
00189                 {
00190                     if (searchParamID == curParam)
00191                     {
00192                         foundParamMappings.push_back(std::make_pair(i, curSlotIndex));
00193                     }
00194
00195                     --numIDsRemaining;
00196                 }
00197             }
00198         }
00199     }
00200
00201     return foundParamMappings;
00202 }
00203
00208     template <class T>
00209     inline void ClearMappedParameterByID(T& ioTable, AAX_CParamID inParameterID)
00210     {

```

```

00211         const auto paramMappings(AAX::FindParameterMappingsInPageTable(ioTable, inParameterID));
00212         for (const auto& locationPair : paramMappings)
00213         {
00214             ioTable.ClearMappedParameter(locationPair.first, locationPair.second);
00215         }
00216     }
00217 }
00218
00219 #endif

```

15.260 AAX_PopStructAlignment.h File Reference

15.260.1 Description

Resets (pops) the struct alignment to its previous value.

See also

AAX_ALIGN_HOST
AAX_ALIGN_ALG
AAX_ALIGN_RESET

Note

Inclusion of this file is mandatory after any 'push' inclusion.

Some compilers do not properly "pop" alignment, so nesting push/pop inclusions is not allowed.

See also

[AAX_Push2ByteStructAlignment.h](#)
[AAX_Push4ByteStructAlignment.h](#)
[AAX_Push8ByteStructAlignment.h](#)

15.261 AAX_PopStructAlignment.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00037 /*=====*/
00038
00039 // Nesting of struct alignment headers is not allowed
00040 #ifndef __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00041 #error "No AAX struct alignment has been set. Cannot undo."
00042 #else
00043 #undef __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00044 #endif
00045
00046 #ifdef _TMS320C6X
00047 // Do nothing for TI
00048 #elif defined (_MSC_VER)

```

```

00049 #pragma pack(pop)
00050 #elif defined (__GNUC__)
00051 // Uncomment this warning suppression if you really want to apply packing to a virtual data
00052 // structure, but note that there is no guarantee of cross-platform compatibility for such
00053 // a structure. For more information, see the AAX_ALIGN_FILE_ALG macro documentation
00054 // #ifdef __clang__
00055 //     #pragma clang diagnostic push
00056 //     #pragma clang diagnostic ignored "-Wno-incompatible-ms-struct"
00057 // #endif
00058 #pragma ms_struct off
00059 // #ifdef __clang__
00060 //     #pragma clang diagnostic pop
00061 // #endif
00062 #pragma pack(pop)
00063 #elif defined (__MWERKS__)
00064 #pragma options align=reset
00065 #else
00066 #error "You need to supply a pragma here to pop structure packing"
00067 #endif

```

15.262 AAX_PostStructAlignmentHelper.h File Reference

15.262.1 Description

Helper file for data alignment macros.

15.263 AAX_PostStructAlignmentHelper.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00018 /*=====*/
00019
00020 #if defined (__clang__)
00021     #if defined (AAX_SDK__PRE_STRUCT_ALIGNMENT_HELPER_DID_PUSH_A_CHANGE)
00022         #pragma clang diagnostic pop
00023     #undef AAX_SDK__PRE_STRUCT_ALIGNMENT_HELPER_DID_PUSH_A_CHANGE
00024     #endif
00025 #endif

```

15.264 AAX_PreStructAlignmentHelper.h File Reference

15.264.1 Description

Helper file for data alignment macros.

15.265 AAX_PreStructAlignmentHelper.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00018 /*=====*/
00019
00020 #if defined (AAX_SDK__PRE_STRUCT_ALIGNMENT_HELPER_DID_PUSH_A_CHANGE)
00021     #warning nested struct alignment directives are not tested
00022 #endif
00023
00024 #if defined (__clang__)
00025     #if __has_warning ("-Wpragma-pack")
00026         #define AAX_SDK__PRE_STRUCT_ALIGNMENT_HELPER_DID_PUSH_A_CHANGE 1
00027         #pragma clang diagnostic push
00028         #pragma clang diagnostic ignored "-Wpragma-pack"
00029     #endif
00030 #endif
```

15.266 AAX_Properties.h File Reference

```
#include "AAX.h"
```

15.266.1 Description

Contains IDs for properties that can be added to an [AAX_IPropertyMap](#).

Enumerations

- enum [AAX_EProperty](#) : int32_t {
 - [AAX_eProperty_NoID](#) = 0 ,
 - [AAX_eProperty_MinProp](#) = 10 ,
 - [AAX_eProperty_PluginSpecPropsBase](#) = 10 ,
 - [AAX_eProperty_ManufacturerID](#) = 11 ,
 - [AAX_eProperty_ProductID](#) = 12 ,
 - [AAX_eProperty_PluginID_Native](#) = 13 ,
 - [AAX_eProperty_PluginID_RTAS](#) = [AAX_eProperty_PluginID_Native](#) ,
 - [AAX_eProperty_PluginID_AudioSuite](#) = 14 ,
 - [AAX_eProperty_PluginID_TI](#) = 15 ,
 - [AAX_eProperty_PluginID_NoProcessing](#) = 16 ,
 - [AAX_eProperty_PluginID_Deprecated](#) = 18 ,
 - [AAX_eProperty_Deprecated_Plugin_List](#) = 21 ,
 - [AAX_eProperty_Related_DSP_Plugin_List](#) = 22 ,
 - [AAX_eProperty_Related_Native_Plugin_List](#) = 23 ,
 - [AAX_eProperty_Deprecated_DSP_Plugin_List](#) = 24 ,
 - [AAX_eProperty_Deprecated_Native_Plugin_List](#) = [AAX_eProperty_Deprecated_Plugin_List](#) ,
 - [AAX_eProperty_PluginID_ExternalProcessor](#) = 25 ,
 - [AAX_eProperty_ExternalProcessorTypeID](#) = 26 ,
 - [AAX_eProperty_ProcessProcPropsBase](#) = 35 ,

```
AAX_eProperty_NativeProcessProc = 36 ,
AAX_eProperty_NativeInstanceInitProc = 37 ,
AAX_eProperty_NativeBackgroundProc = 38 ,
AAX_eProperty_TIDLLFileName = 39 ,
AAX_eProperty_TIProcessProc = 40 ,
AAX_eProperty_TIInstanceInitProc = 41 ,
AAX_eProperty_TIBackgroundProc = 42 ,
AAX_eProperty_GeneralPropsBase = 50 ,
AAX_eProperty_InputStemFormat = 51 ,
AAX_eProperty_OutputStemFormat = 52 ,
AAX_eProperty_DSP_AudioBufferLength = 54 ,
AAX_eProperty_AudioBufferLength = AAX_eProperty_DSP_AudioBufferLength ,
AAX_eProperty_LatencyContribution = 56 ,
AAX_eProperty_SampleRate = 58 ,
AAX_eProperty_CanBypass = 60 ,
AAX_eProperty_SideChainStemFormat = 61 ,
AAX_eProperty_TI_SharedCycleCount = 62 ,
AAX_eProperty_TI_InstanceCycleCount = 63 ,
AAX_eProperty_TI_MaxInstancesPerChip = 64 ,
AAX_eProperty_TI_ForceAllowChipSharing = 65 ,
AAX_eProperty_AlwaysBypass = 75 ,
AAX_eProperty_ShowInMenus = 76 ,
AAX_eProperty_HybridOutputStemFormat = 90 ,
AAX_eProperty_HybridInputStemFormat = 91 ,
AAX_eProperty_AudiosuitePropsBase = 100 ,
AAX_eProperty_UsesRandomAccess = 101 ,
AAX_eProperty_RequiresAnalysis = 102 ,
AAX_eProperty_OptionalAnalysis = 103 ,
AAX_eProperty_AllowPreviewWithoutAnalysis = 104 ,
AAX_eProperty_DestinationTrack = 105 ,
AAX_eProperty_RequestsAllTrackData = 106 ,
AAX_eProperty_ContinuousOnly = 107 ,
AAX_eProperty_MultiInputModeOnly = 108 ,
AAX_eProperty_DisablePreview = 110 ,
AAX_eProperty_DoesntIncrOutputSample = 112 ,
AAX_eProperty_NumberOfInputs = 113 ,
AAX_eProperty_NumberOfOutputs = 114 ,
AAX_eProperty_DisableHandles = 115 ,
AAX_eProperty_SupportsSideChainInput = 116 ,
AAX_eProperty_NeedsOutputDithered = 117 ,
AAX_eProperty_DisableAudiosuiteReverse = 118 ,
AAX_eProperty_MaxASProp ,
AAX_eProperty_GUIBase = 150 ,
AAX_eProperty_UsesClientGUI = 151 ,
AAX_eProperty_MaxGUIProp ,
AAX_eProperty_MeterBase = 199 ,
AAX_eProperty_Meter_Type = 200 ,
AAX_eProperty_Meter_Orientation = 201 ,
AAX_eProperty_Meter_Ballistics = 202 ,
AAX_eProperty_MaxMeterProp ,
AAX_eProperty_ConstraintBase = 299 ,
AAX_eProperty_Constraint_Location = 300 ,
AAX_eProperty_Constraint_Topology = 301 ,
AAX_eProperty_Constraint_NeverUnload = 302 ,
AAX_eProperty_Constraint_NeverCache = 303 ,
AAX_eProperty_Constraint_MultiMonoSupport = 304 ,
AAX_eProperty_MaxConstraintProp ,
AAX_eProperty_FeaturesBase = 305 ,
```

```

AAX_eProperty_SupportsSaveRestore = 305 ,
AAX_eProperty_UsesTransport = 306 ,
AAX_eProperty_StoreXMLPageTablesByEffect = 307 ,
AAX_eProperty_StoreXMLPageTablesByType = AAX_eProperty_StoreXMLPageTablesByEffect ,
AAX_eProperty_RequiresChunkCallsOnMainThread = 308 ,
AAX_eProperty_ObservesTransportState = 309 ,
AAX_eProperty_UsesTransportControl = 311 ,
AAX_eProperty_MaxFeaturesProp ,
AAX_eProperty_ConstraintBase_2 = 350 ,
AAX_eProperty_Constraint_AlwaysProcess = 351 ,
AAX_eProperty_Constraint_DoNotApplyDefaultSettings = 352 ,
AAX_eProperty_MaxConstraintProp_2 ,
AAX_eProperty_DebugPropertiesBase = 400 ,
AAX_eProperty_EnableHostDebugLogs = 401 ,
AAX_eProperty_MaxProp ,
AAX_eProperty_MaxCap = 10000 }

```

The list of properties that can be added to an [AAX_IPropertyMap](#).

Functions

- [AAX_ENUM_SIZE_CHECK](#) ([AAX_EProperty](#))

15.266.2 Enumeration Type Documentation

15.266.2.1 AAX_EProperty

```
enum AAX\_EProperty : int32_t
```

The list of properties that can be added to an [AAX_IPropertyMap](#).

See [AAX_IPropertyMap::AddProperty\(\)](#) for more information

Sections

- [Plug-In spec properties](#)
- [ProcessProc properties](#)
- [General properties](#)
- [TI-specific properties](#)
- [Offline \(AudioSuite\) properties](#)
- [GUI properties](#)
- [Meter properties](#)
- [Plug-in management constraints](#)

Legacy Porting Notes These property IDs are somewhat analogous to the pluginGestalt system in the legacy SDK, and several [AAX_EProperty](#) values correlate directly with a corresponding legacy plug-in gestalt.

Legacy Porting Notes To ensure session interchange compatibility, make sure the 4 character IDs for [AAX_eProperty_ManufacturerID](#), [AAX_eProperty_ProductID](#), [AAX_eProperty_PluginID_Native](#), and [AAX_eProperty_PluginID_AudioSuite](#) are identical to the legacy SDK's counterpart.

Enumerator

AAX_eProperty_NoID	
AAX_eProperty_MinProp	
AAX_eProperty_PluginSpecPropsBase	
AAX_eProperty_ManufacturerID	<p>Four-character osid-style manufacturer identifier. Should be registered with Avid, and must be identical for all plug-ins from the same manufacturer.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level for plug-ins that support audio processing using a ProcessProc callback, or at the Effect level for all other plug-ins. <p>Legacy Porting Notes For legacy plug-in session compatibility, this ID should match the Manufacturer ID used in the corresponding legacy plug-ins.</p>
AAX_eProperty_ProductID	<p>Four-character osid-style Effect identifier. Must be identical for all ProcessProcs within a single Effect.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level for plug-ins that support audio processing using a ProcessProc callback, or at the Effect level for all other plug-ins. <p>Legacy Porting Notes For legacy plug-in session compatibility, this ID should match the Product ID used in the corresponding legacy plug-in.</p>

Enumerator

AAX_eProperty_PluginID_Native	<p>Four-character osid-style plug-in type identifier for real-time native audio Effects. All registered plug-in type IDs (AAX_eProperty_PluginID_Native, AAX_eProperty_PluginID_AudioSuite, AAX_eProperty_PluginID_TI, etc.) must be unique across all ProcessProcs registered within a single Effect.</p> <p>Warning</p> <p>As with all plug-in ID properties, this value must remain constant across all releases of the plug-in which support this Effect configuration. The value of this property should be stored in a constant rather than being calculated at run-time in order to avoid unresolvable compatibility issues with saved sessions which can occur if an ID value is accidentally changed between two plug-in version releases.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>Legacy Porting Notes For legacy plug-in session compatibility, this ID should match the Type ID used in the corresponding legacy RTAS plug-in Types.</p>
AAX_eProperty_PluginID_RTAS	<p>Deprecated Use AAX_eProperty_PluginID_Native</p>
AAX_eProperty_PluginID_AudioSuite	<p>Four-character osid-style plug-in type identifier for offline native audio Effects. All registered plug-in type IDs (AAX_eProperty_PluginID_Native, AAX_eProperty_PluginID_AudioSuite, AAX_eProperty_PluginID_TI, etc.) must be unique across all ProcessProcs registered within a single Effect.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level for plug-ins that support audio processing using a ProcessProc callback, or at the Effect level for all other AudioSuite plug-ins (e.g. those that use the AAX_IHostProcessor interface.) <p>Legacy Porting Notes For legacy plug-in session compatibility, this ID should match the Type ID used in the corresponding legacy AudioSuite plug-in Types.</p>

Enumerator

AAX_eProperty_PluginID_TI	<p>Four-character osid-style plug-in type identifier for real-time TI-accelerated audio Effect types. All registered plug-in type IDs (AAX_eProperty_PluginID_Native, AAX_eProperty_PluginID_AudioSuite, AAX_eProperty_PluginID_TI, etc.) must be unique across all ProcessProcs registered within a single Effect.</p> <p>Warning</p> <p>As with all plug-in ID properties, this value must remain constant across all releases of the plug-in which support this Effect configuration. The value of this property should be stored in a constant rather than being calculated at run-time in order to avoid unresolvable compatibility issues with saved sessions which can occur of an ID value is accidentally changed between two plug-in version releases.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>Legacy Porting Notes For legacy plug-in session compatibility, this ID should match the Type ID used in the corresponding legacy TDM plug-in Types.</p>
AAX_eProperty_PluginID_NoProcessing	<p>Four-character osid-style plug-in type identifier for Effect types that do not process audio. All registered plug-in type IDs (AAX_eProperty_PluginID_Native, AAX_eProperty_PluginID_AudioSuite, AAX_eProperty_PluginID_TI, etc.) must be unique across all ProcessProcs registered within a single Effect.</p> <p>Warning</p> <p>As with all plug-in ID properties, this value must remain constant across all releases of the plug-in which support this Effect configuration. The value of this property should be stored in a constant rather than being calculated at run-time in order to avoid unresolvable compatibility issues with saved sessions which can occur of an ID value is accidentally changed between two plug-in version releases.</p> <ul style="list-style-type: none"> • Apply this property at the Effect level

Enumerator

AAX_eProperty_PluginID_Deprecated	<p>Four-character osid-style plug-in type identifier for a corresponding deprecated type. Only one deprecated effect ID may correspond to each valid (non-deprecated) effect ID. To associate a plug-in type with more than one deprecated type, use the following properties instead:</p> <ul style="list-style-type: none"> • AAX_eProperty_Deprecated_DSP_Plugin_List • AAX_eProperty_Deprecated_Native_Plugin_List • Apply this property at the ProcessProc level
AAX_eProperty_Deprecated_Plugin_List	<p>Deprecated Use AAX_eProperty_Deprecated_Native_Plugin_List and AAX_eProperty_Deprecated_DSP_Plugin_List. See AAX_eProperty_PluginID_RTAS for an example.</p>
AAX_eProperty_Related_DSP_Plugin_List	<p>Specify a list of DSP plug-ins that are related to a plug-in type.</p> <ul style="list-style-type: none"> • For example, use this property inside a Native process to tell the host that this plug-in can be used in place of a DSP version. • This property must be applied at the ProcessProc level and used with the AAX_IPropertyMap::AddPropertyWithIDArray method, which takes a list of full plug-in identifier specification triads (ManufacturerID, ProductID, PluginID)
AAX_eProperty_Related_Native_Plugin_List	<p>Specify a list of Native plug-ins that are related to a plug-in type.</p> <ul style="list-style-type: none"> • This property must be applied at the ProcessProc level and used with the AAX_IPropertyMap::AddPropertyWithIDArray method, which takes a list of full plug-in identifier specification triads (ManufacturerID, ProductID, PluginID)
AAX_eProperty_Deprecated_DSP_Plugin_List	<p>Specify a list of DSP plug-ins that are deprecated by a new plug-in type.</p> <ul style="list-style-type: none"> • This property must be applied at the ProcessProc level and used with the AddPropertyWithIDArray, which is a list of full plug-in specs (ManufacturerID, ProductID, PluginID)

Enumerator

AAX_eProperty_Deprecated_Native_Plugin_List	<p>Specify a list of Native plug-ins that are deprecated by a new plug-in type.</p> <ul style="list-style-type: none"> This property must be applied at the ProcessProc level and used with the AddPropertyWithIDArray, which is a list of full plug-in specs (ManufacturerID, ProductID, PluginID)
AAX_eProperty_PluginID_ExternalProcessor	<p>Four-character osid-style plug-in type identifier for audio effects rendered on external hardware.</p> <p>Note</p> <p>This property is not currently used by any AAX plug-in host software</p> <p>All registered plug-in type IDs must be unique across all ProcessProcs registered within a single Effect.</p> <p>Warning</p> <p>As with all plug-in ID properties, this value must remain constant across all releases of the plug-in which support this Effect configuration. The value of this property should be stored in a constant rather than being calculated at run-time in order to avoid unresolvable compatibility issues with saved sessions which can occur if an ID value is accidentally changed between two plug-in version releases.</p> <ul style="list-style-type: none"> Apply this property at the ProcessProc level
AAX_eProperty_ExternalProcessorTypeID	<p>Identifier for the type of the external processor hardware.</p> <p>See also</p> <p>AAX_eProperty_PluginID_ExternalProcessor</p> <p>The value of this property will be specific to the external processor hardware. Currently there are no public external processor hardware type IDs.</p> <ul style="list-style-type: none"> Apply this property at the ProcessProc level
AAX_eProperty_ProcessProcPropsBase	
AAX_eProperty_NativeProcessProc	<p>Address of a native effect's ProcessProc callback</p> <p>Data type: AAX_CProcessProc</p> <p>For use with AAX_IComponentDescriptor::AddProcessProc()</p>
AAX_eProperty_NativeInstanceInitProc	<p>Address of a native effect's instance initialization callback</p> <p>Data type: AAX_CInstanceInitProc</p> <p>For use with AAX_IComponentDescriptor::AddProcessProc()</p>

Enumerator

AAX_eProperty_NativeBackgroundProc	Address of a native effect's background callback Data type: AAX_CBackgroundProc For use with AAX_IComponentDescriptor::AddProcessProc()
AAX_eProperty_TIDLLFileName	Name of the DLL for a TI effect Data type: UTF-8 C-string For use with AAX_IComponentDescriptor::AddProcessProc()
AAX_eProperty_TIProcessProc	Name of a TI effect's ProcessProc callback Data type: C-string For use with AAX_IComponentDescriptor::AddProcessProc()
AAX_eProperty_TIInstanceInitProc	Name of a TI effect's instance initialization callback Data type: C-string For use with AAX_IComponentDescriptor::AddProcessProc()
AAX_eProperty_TIBackgroundProc	Name of a TI effect's background callback Data type: C-string For use with AAX_IComponentDescriptor::AddProcessProc()
AAX_eProperty_GeneralPropsBase	_____
AAX_eProperty_InputStemFormat	Input stem format. One of AAX_EStemFormat . • Apply this property at the ProcessProc level For offline processing, use AAX_eProperty_NumberOfInputs
AAX_eProperty_OutputStemFormat	Output stem format. One of AAX_EStemFormat . • Apply this property at the ProcessProc level For offline processing, use AAX_eProperty_NumberOfOutputs
AAX_eProperty_DSP_AudioBufferLength	Audio buffer length for DSP processing callbacks. One of AAX_EAudioBufferLengthDSP . • Apply this property at the ProcessProc level • This property is only applicable to DSP algorithms
AAX_eProperty_AudioBufferLength	Deprecated Use AAX_eProperty_DSP_AudioBufferLength

Enumerator

AAX_eProperty_LatencyContribution	<p>Default latency contribution of a given processing callback, in samples.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>Unlike most properties, an Effect's latency contribution may also be changed dynamically at runtime. This is done via AAX_IController::SetSignalLatency(). Dynamic latency reporting may not be recognized by the host application in all circumstances, however, so Effects should <i>always</i> define any nonzero initial latency value using AAX_eProperty_LatencyContribution</p> <p>Host Compatibility Notes Maximum delay compensation limits will vary from host to host. If your plug-in exceeds the delay compensation sample limit for a given AAX host then you should note this limitation in your user documentation. Example limits:</p> <ul style="list-style-type: none"> • Pro Tools 9 and higher: 16,383 samples at 44.1/48 kHz, 32,767 samples at 88.2/96 kHz, or 65,534 samples at 176.4/192 kHz • Media Composer 8.1 and higher: 16,383 samples at 44.1/48 kHz, 32,767 samples at 88.2/96 kHz
AAX_eProperty_SampleRate	<p>Specifies which sample rates the Effect supports. A mask of AAX_ESampleRateMask.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>See also</p> <p>AAX_IComponentDescriptor::AddSampleRate()</p>

Enumerator

AAX_eProperty_CanBypass	<p>The plug-in supports a Master Bypass control.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>Nearly all AAX plug-ins should set this property to <code>true</code></p> <p>Set this property to <code>false</code> (0) to disable Master Bypass for plug-ins that cannot be bypassed, such as fold-down plug-ins that convert to a narrower channel format.</p> <p>Legacy Porting Notes Was <code>pluginGestalt_CanBypass</code>.</p>
AAX_eProperty_SideChainStemFormat	<p>Side chain stem format. One of AAX_EStemFormat.</p> <p>Host Compatibility Notes Currently Pro Tools supports only AAX_eStemFormat_Mono side chain inputs</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>Host Compatibility Notes <code>AAX_eProperty_SideChainStemFormat</code> is not currently implemented in DAE or AAE</p>
AAX_eProperty_TI_SharedCycleCount	<p>Shared cycle count (outer, per clump, loop overhead)</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level • This property is only applicable to DSP algorithms
AAX_eProperty_TI_InstanceCycleCount	<p>Instance cycle count (inner, per instance, loop overhead)</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level • This property is only applicable to DSP algorithms
AAX_eProperty_TI_MaxInstancesPerChip	<p>Maximum number of instances of this plug-in that can be loaded on a chip. This property is only used for DMA and background thread-enabled plug-ins.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level • This property is only applicable to DSP algorithms

Enumerator

AAX_eProperty_TI_ForceAllowChipSharing	<p>Allow different plug-in types to share the same DSP even if AAX_eProperty_TI_MaxInstancesPerChip is declared. In general, this is not desired behavior. However, this can be useful if your plug-in instance counts are bound by a system constraint other than CPU usage and you require chip-sharing between instances of different types of the plug-in.</p> <p>Note</p> <p>In addition to defining this property, the types which will share allocations on the same DSP chip must be compiled into the same ELF DLL file.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level • This property is only applicable to DSP algorithms
AAX_eProperty_AlwaysBypass	<p>The plug-in never alters its audio signal, audio output is always equal to audio input.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>Setting this property allows host to optimize audio routing and reduce audio latency.</p>
AAX_eProperty_ShowInMenus	<p>Indicates whether or not the plug-in should be shown in insert menus.</p> <ul style="list-style-type: none"> • Apply this property to show or hide the plug-in from the Pro Tools insert menus. • This property value is <code>true</code> by default.
AAX_eProperty_HybridOutputStemFormat	<p>Hybrid Output stem format. One of AAX_EStemFormat. This property represents the stem format for the audio channels that are sent from the ProcessProc callback to the AAX_IEffectParameters::RenderAudio_Hybrid() method</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level • Normally plugins will set this to the same thing as AAX_eProperty_InputStemFormat
AAX_eProperty_HybridInputStemFormat	<p>Hybrid Input stem format. One of AAX_EStemFormat. This property represents the stem format for the audio channels that are sent from the AAX_IEffectParameters::RenderAudio_Hybrid() method to the ProcessProc callback</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level • Normally plugins will set this to the same thing as AAX_eProperty_OutputStemFormat
AAX_eProperty_AudiosuitePropsBase	

Enumerator

AAX_eProperty_UsesRandomAccess	<p>The Effect requires random access to audio data.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to Host Processor algorithms <p>Legacy Porting Notes Was pluginGestalt_Uses↔ RandomAccess</p>
AAX_eProperty_RequiresAnalysis	<p>The Effect requires an analysis pass.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Legacy Porting Notes Was pluginGestalt_↔ RequiresAnalysis</p>
AAX_eProperty_OptionalAnalysis	<p>The Effect supports an analysis pass, but does not require it.</p> <p>Host Compatibility Notes In Media Composer, optional analysis will also be performed automatically before each channel is rendered. See MCDEV-2904</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Legacy Porting Notes Was pluginGestalt_↔ OptionalAnalysis</p>
AAX_eProperty_AllowPreviewWithoutAnalysis	<p>The Effect requires analysis, but is also allowed to preview.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Legacy Porting Notes Was pluginGestalt_↔ AnalyzeOnTheFly</p>

Enumerator

AAX_eProperty_DestinationTrack	<p>Informs the host application to reassign output to a different track.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Host Compatibility Notes This property is not supported on Media Composer</p> <p>Legacy Porting Notes Was pluginGestalt_↔ DestinationTrack</p>
AAX_eProperty_RequestsAllTrackData	<p>The host should make all of the processed track's data available to the Effect.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to Host Processor algorithms <p>Legacy Porting Notes Was pluginGestalt_↔ RequestsAllTrackData</p>
AAX_eProperty_ContinuousOnly	<p>The Effect only processes on continuous data and does not support 'clip by clip' rendering.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Legacy Porting Notes Was pluginGestalt_↔ ContinuousOnly</p>
AAX_eProperty_MultiInputModeOnly	<p>The Effect wants multi-input mode only (no mono mode option)</p> <p>Note</p> <p>See bug PT-258560 / PT-256919</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Legacy Porting Notes Was pluginGestalt_Multi↔ InputModeOnly</p>

Enumerator

AAX_eProperty_DisablePreview	<p>The Effect does not support preview.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Legacy Porting Notes Was pluginGestalt_Disable↔ Preview</p>
AAX_eProperty_DoesntIncrOutputSample	<p>The Effect may not increment its output sample during some rendering calls.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to Host Processor algorithms <p>Legacy Porting Notes Was pluginGestalt_Doesnt↔ IncrOutputSample</p>
AAX_eProperty_NumberOfInputs	<p>The number of input channels that the plug-in supports.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to Host Processor algorithms <p>For real-time processing, use AAX_eProperty_InputStemFormat</p>
AAX_eProperty_NumberOfOutputs	<p>The number of output channels that the plug-in supports.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to Host Processor algorithms <p>For real-time processing, use AAX_eProperty_OutputStemFormat</p>
AAX_eProperty_DisableHandles	<p>Prevents the application of rendered region handles by the host.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing

Enumerator

AAX_eProperty_SupportsSideChainInput	<p>Tells the host that the plug-in supports side chain inputs.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing
AAX_eProperty_NeedsOutputDithered	<p>Requests that the host apply dithering to the Effect's output.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Legacy Porting Notes Was pluginGestalt_Needs↔OutputDithered</p>
AAX_eProperty_DisableAudiosuiteReverse	<p>The plug-in supports audiosuite reverse. By default, all reverb and delay plug-ins support this feature. If a plug-in needs to opt out of this feature, they can set this property to true.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level • This property is only applicable to offline processing <p>Host Compatibility Notes AAX_eProperty_↔DisableAudiosuite↔Reverse is not currently implemented</p>
AAX_eProperty_MaxASProp	
AAX_eProperty_GUIBase	
AAX_eProperty_UsesClientGUI	<p>Requests a host-generated GUI based on the Effect's parameters. Use this property while your plug-in is in development to test the plug-in's data model and algorithm before its GUI has been created, or when troubleshooting problems to isolate the data model and algorithm operation from the plug-in's GUI.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level <p>Host Compatibility Notes Currently supported by Pro Tools only</p> <p>Note</p> <p>See PTSW-189725 / PT-218397</p>
AAX_eProperty_MaxGUIProp	
AAX_eProperty_MeterBase	

Enumerator

AAX_eProperty_Meter_Type	<p>Indicates meter type as one of AAX_EMeterType.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor::AddMeterDescription() level
AAX_eProperty_Meter_Orientation	<p>Indicates meter orientation as one of AAX_EMeterOrientation.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor::AddMeterDescription() level
AAX_eProperty_Meter_Ballistics	<p>Indicates meter ballistics preference as one of AAX_EMeterBallisticType.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor::AddMeterDescription() level
AAX_eProperty_MaxMeterProp	
AAX_eProperty_ConstraintBase	
AAX_eProperty_Constraint_Location	<p>Constraint on the algorithm's location, as a mask of AAX_EConstraintLocationMask.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level
AAX_eProperty_Constraint_Topology	<p>Constraint on the topology of the Effect's modules, as one of AAX_EConstraintTopology.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level
AAX_eProperty_Constraint_NeverUnload	<p>Tells the host that it should never unload the plug-in binary.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level <p>Host Compatibility Notes AAX_eProperty_↔ Constraint_NeverUnload is not currently implemented in DAE or AAE</p>

Enumerator

AAX_eProperty_Constraint_NeverCache	<p>Tells the host that it should never cache the plug-in binary. Only use this if required as there is a performance penalty on launch to not use the Cache. Set this property to 1, if you really need to not cache. Default is 0. The most common reason for a plug-in to require this constraint is if the plug-in's configuration can change based on external conditions. Most of the data contained in the plug-in's description routine is cached, so if the plug-in description can change between launches of the host application then the plug-in should apply this constraint to prevent the host from using stale description information. This property should be applied at the collection level as it affects the entire bundle.</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level
AAX_eProperty_Constraint_MultiMonoSupport	<p>Indicates whether or not the plug-in supports multi-mono configurations (<code>true/false</code>)</p> <p>Note</p> <p>Multi-mono mode may not work as expected for VIs and other plug-ins which rely on non-global MIDI input. Depending on the host, multi-mono instances may not all be automatically connected to the same MIDI port upon instantiation. Therefore it is recommended to set this property to 0 for any plug-ins if this lack of automatic connection may confuse users.</p> <ul style="list-style-type: none"> • Apply this property at the ProcessProc level
AAX_eProperty_MaxConstraintProp	
AAX_eProperty_FeaturesBase	
AAX_eProperty_SupportsSaveRestore	<p>Indicates whether or not the plug-in supports Save/Restore features. (<code>true/false</code>)</p> <ul style="list-style-type: none"> • Apply this property to show or hide the Settings section in the plug-in window. • This property value is true by default. <p>Legacy Porting Notes Was <code>pluginGestalt_↔ SupportsSaveRestore</code></p>
AAX_eProperty_UsesTransport	<p>Indicates whether or not the plug-in uses transport requests. (<code>true/false</code>)</p> <ul style="list-style-type: none"> • Apply this property if your plug-in uses AAX_ITransport class. • Apply this property at the AAX_IEffectDescriptor level

Enumerator

AAX_eProperty_StoreXMLPageTablesByEffect	<p>This property specifies whether the plug-in bundle contains an XML file per plug-in type. AAX plug-ins always provide XML page table data via external files referenced by AAX_eResourceType_PageTable. If AAX_eProperty_StoreXMLPageTablesByEffect is not defined or is set to 0 (the default) then the host may assume that all Effects in the collection use the same XML page table file. If this property is set to a non-zero value, the plug-in may describe a different AAX_eResourceType_PageTable for each separate Effect.</p> <p>This property needs to be set at the collection level.</p>
AAX_eProperty_StoreXMLPageTablesByType	<p>Deprecated Use AAX_eProperty_StoreXMLPageTablesByEffect</p>
AAX_eProperty_RequiresChunkCallsOnMainThread	<p>Indicates whether the plug-in supports SetChunk and GetChunk calls on threads other than the main thread. It is actually important for plug-ins to support these calls on non-main threads, so that is the default. However, in response to a few companies having issues with this, we have decided to support this constraint for now. property value should be set to true if you need Chunk calls on the main thread. Values: 0 (off, default), 1 (on)</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level
AAX_eProperty_ObservesTransportState	<p>Indicates whether the plug-in subscribes to the TransportStateChanged notification to receive transport info. property value should be set to true if you need subscribe to the TransportStateNotification. Values: 0 (off, default), 1 (on)</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level
AAX_eProperty_UsesTransportControl	<p>Indicates whether or not the plug-in uses transport control requests. (true/false)</p> <ul style="list-style-type: none"> • Apply this property if your plug-in uses AAX_IACFTransportControl methods in the AAX_ITransport class. • Apply this property at the AAX_IEffectDescriptor level
AAX_eProperty_MaxFeaturesProp	
AAX_eProperty_ConstraintBase_2	

Enumerator

AAX_eProperty_Constraint_AlwaysProcess	<p>Indicates that the plug-in's processing should never be disabled by the host (<code>true/false</code>) Some hosts will disable processing for plug-in chains in certain circumstances to conserve system resources, e.g. when the chains' output drops to silence for an extended period.</p> <p>Note</p> <p>This property may impact performance of other plug-ins. For example, the Dynamic Plug-In Processing feature in Pro Tools operates over chains of plug-ins rather than single instances; any plug-in that defines AAX_eProperty_Constraint_AlwaysProcess will force its entire signal chain to continue processing. Therefore it is important to avoid using this property unless features such as Dynamic Plug-In Processing are actually interfering in some way with the operation of the plug-in.</p> <ul style="list-style-type: none"> • This property value is false by default. • Apply this property at the AAX_IEffectDescriptor level
AAX_eProperty_Constraint_DoNotApplyDefault↔ Settings	<p>Requests that the host does not send default settings chunks to the plug-in after instantiation (<code>true/false</code>) Some hosts will apply the plug-in's default settings via chunks after creating a new plug-in instance as a way to ensure that the all new plug-in instances are initialized to the same state. If a plug-in can make this guarantee itself and does not wish to receive any default settings chunks from the host after instantiation then it may set this property.</p> <p>Support for this property is not guaranteed; the plug-in must be able to handle default settings chunk application even if this property is set, or clearly document the plug-in's host compatibility.</p> <p>Note</p> <p>See bug PT-284916</p> <ul style="list-style-type: none"> • Apply this property at the AAX_IEffectDescriptor level #
AAX_eProperty_MaxConstraintProp_2	
AAX_eProperty_DebugPropertiesBase	

Enumerator

AAX_eProperty_EnableHostDebugLogs	<p>Enables host debug logging for this plug-in. This logging is made via DigiTrace using the DTF_AAXHOST facility, generally at DTP_LOW priority</p> <ul style="list-style-type: none"> • It is recommended to set this property to 1 for debug builds and to 0 for release builds of a plug-in • Apply this property at the AAX_IEffectDescriptor level
AAX_eProperty_MaxProp	
AAX_eProperty_MaxCap	

15.266.3 Function Documentation

15.266.3.1 AAX_ENUM_SIZE_CHECK()

```
AAX_ENUM_SIZE_CHECK (
    AAX_EProperty )
```

15.267 AAX_Properties.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00023 #pragma once
00024 #ifndef AAX_PROPERTIES_H
00025 #define AAX_PROPERTIES_H
00027
00028 #ifndef _AAX_H_
00029 #include "AAX.h"
00030 #endif
00031
00032
00033 // Add new values only at the end of existing sections!
00034
00063 // Current CCS doesn't support C++11
00064 #ifdef _TMS320C6X
00065 enum AAX_EProperty
00066 #else
00067 enum AAX_EProperty : int32_t
00068 #endif
00069 {
```

```

00070     AAX_eProperty_NoID = 0,
00071     AAX_eProperty_MinProp = 10, // MUST BE EQUAL TO MINIMUM PROPERTY VALUE
00072
00073 //-----
00074 #if 0
00075 #pragma mark Plug-In spec properties
00076 #endif
00082     AAX_eProperty_PlugInSpecPropsBase = 10,
00095     AAX_eProperty_ManufacturerID = 11,
00108     AAX_eProperty_ProductID = 12,
00127     AAX_eProperty_PlugInID_Native = 13,
00130     AAX_eProperty_PlugInID_RTAS = AAX_eProperty_PlugInID_Native,
00145     AAX_eProperty_PlugInID_AudioSuite = 14,
00165     AAX_eProperty_PlugInID_TI = 15,
00182     AAX_eProperty_PlugInID_NoProcessing = 16,
00192     AAX_eProperty_PlugInID_Deprecated = 18,
00196     AAX_eProperty_Deprecated_Plugin_List = 21,
00205     AAX_eProperty_Related_DSP_Plugin_List = 22,
00212     AAX_eProperty_Related_Native_Plugin_List = 23,
00218     AAX_eProperty_Deprecated_DSP_Plugin_List = 24,
00224     AAX_eProperty_Deprecated_Native_Plugin_List = AAX_eProperty_Deprecated_Plugin_List,
00241     AAX_eProperty_PlugInID_ExternalProcessor = 25,
00251     AAX_eProperty_ExternalProcessorTypeID = 26,
00253
00254 //-----
00255 #if 0
00256 #pragma mark ProcessProc properties
00257 #endif
00263     AAX_eProperty_ProcessProcPropsBase = 35,
00270     AAX_eProperty_NativeProcessProc = 36,
00277     AAX_eProperty_NativeInstanceInitProc = 37,
00284     AAX_eProperty_NativeBackgroundProc = 38,
00291     AAX_eProperty_TIDLLFileName = 39,
00298     AAX_eProperty_TIProcessProc = 40,
00305     AAX_eProperty_TIInstanceInitProc = 41,
00312     AAX_eProperty_TIBackgroundProc = 42,
00314
00315 //-----
00316 #if 0
00317 #pragma mark General properties
00318 #endif
00324     AAX_eProperty_GeneralPropsBase = 50,
00331     AAX_eProperty_InputStemFormat = 51,
00338     AAX_eProperty_OutputStemFormat = 52,
00345     AAX_eProperty_DSP_AudioBufferLength = 54,
00348     AAX_eProperty_AudioBufferLength = AAX_eProperty_DSP_AudioBufferLength,
00365     AAX_eProperty_LatencyContribution = 56,
00372     AAX_eProperty_SampleRate = 58,
00384     AAX_eProperty_CanBypass = 60,
00393     AAX_eProperty_SideChainStemFormat = 61,
00395
00396 //-----
00397 #if 0
00398 #pragma mark TI-specific properties
00399 #endif
00409     AAX_eProperty_TI_SharedCycleCount = 62,
00415     AAX_eProperty_TI_InstanceCycleCount = 63,
00422     AAX_eProperty_TI_MaxInstancesPerChip = 64,
00436     AAX_eProperty_TI_ForceAllowChipSharing = 65,
00438
00439 //-----
00440 #if 0
00441 #pragma mark General properties (continued)
00442 #endif
00455     AAX_eProperty_AlwaysBypass = 75,
00461     AAX_eProperty_ShowInMenus = 76,
00463
00464 //-----
00465 #if 0
00466 #pragma mark AAX Hybrid properties
00467 #endif
00482     AAX_eProperty_HybridOutputStemFormat = 90,
00493     AAX_eProperty_HybridInputStemFormat = 91,
00495
00496 //-----
00497 #if 0
00498 #pragma mark Offline (AudioSuite) properties
00499 #endif
00505     AAX_eProperty_AudiosuitePropsBase = 100,
00514     AAX_eProperty_UsesRandomAccess = 101,
00522     AAX_eProperty_RequiresAnalysis = 102,
00533     AAX_eProperty_OptionalAnalysis = 103,
00541     AAX_eProperty_AllowPreviewWithoutAnalysis = 104,
00551     AAX_eProperty_DestinationTrack = 105,
00560     AAX_eProperty_RequestsAllTrackData = 106,
00569     AAX_eProperty_ContinuousOnly = 107,
00579     AAX_eProperty_MultiInputModeOnly = 108,

```

```

00587     AAX_eProperty_DisablePreview = 110,
00596     AAX_eProperty_DoesntIncrOutputSample = 112,
00605     AAX_eProperty_NumberOfInputs = 113,
00614     AAX_eProperty_NumberOfOutputs = 114,
00620     AAX_eProperty_DisableHandles = 115,
00626     AAX_eProperty_SupportsSideChainInput = 116,
00634     AAX_eProperty_NeedsOutputDithered = 117,
00644     AAX_eProperty_DisableAudiosuiteReverse = 118,
00645
00646     AAX_eProperty_MaxASProp, // Intentionally given no explicit value
00648
00649 //-----
00650 #if 0
00651 #pragma mark GUI properties
00652 #endif
00658     AAX_eProperty_GUIBase = 150,
00671     AAX_eProperty_UsesClientGUI = 151,
00672
00673     AAX_eProperty_MaxGUIProp, // Intentionally given no explicit value
00675
00676 //-----
00677 #if 0
00678 #pragma mark Meter properties
00679 #endif
00688     AAX_eProperty_MeterBase = 199,
00693     AAX_eProperty_Meter_Type = 200,
00698     AAX_eProperty_Meter_Orientation = 201,
00703     AAX_eProperty_Meter_Ballistics = 202,
00704
00705     AAX_eProperty_MaxMeterProp, // Intentionally given no explicit value
00707
00708 //-----
00709 #if 0
00710 #pragma mark Plug-in management constraints
00711 #endif
00720     AAX_eProperty_ConstraintBase = 299,
00725     AAX_eProperty_Constraint_Location = 300,
00731     AAX_eProperty_Constraint_Topology = 301,
00738     AAX_eProperty_Constraint_NeverUnload = 302,
00752     AAX_eProperty_Constraint_NeverCache = 303,
00763     AAX_eProperty_Constraint_MultiMonoSupport = 304,
00764
00765     AAX_eProperty_MaxConstraintProp, // Intentionally given no explicit value
00767
00768 //-----
00769 #if 0
00770 #pragma mark Plug-in features
00771 #endif
00778     AAX_eProperty_FeaturesBase = 305, // No room was given, so this equals
AAX_eProperty_SupportsSaveRestore
00787     AAX_eProperty_SupportsSaveRestore = 305,
00794     AAX_eProperty_UsesTransport = 306,
00806     AAX_eProperty_StoreXMLPageTablesByEffect = 307,
00809     AAX_eProperty_StoreXMLPageTablesByType = AAX_eProperty_StoreXMLPageTablesByEffect,
00821     AAX_eProperty_RequiresChunkCallsOnMainThread = 308,
00831     AAX_eProperty_ObservesTransportState = 309,
00838     AAX_eProperty_UsesTransportControl = 311,
00839
00840     AAX_eProperty_MaxFeaturesProp, // Intentionally given no explicit value
00842
00843 //-----
00844 #if 0
00845 #pragma mark Plug-in management constraints (continued)
00846 #endif
00852     AAX_eProperty_ConstraintBase_2 = 350,
00853
00869     AAX_eProperty_Constraint_AlwaysProcess = 351,
00870
00892     AAX_eProperty_Constraint_DoNotApplyDefaultSettings = 352,
00893
00894     AAX_eProperty_MaxConstraintProp_2, // Intentionally given no explicit value
00896
00897 //-----
00898 #if 0
00899 #pragma mark Debug properties
00900 #endif
00904     AAX_eProperty_DebugPropertiesBase = 400,
00912     AAX_eProperty_EnableHostDebugLogs = 401,
00914
00915     AAX_eProperty_MaxProp, // ALWAYS LEAVE AS LAST PROPERTY VALUE
00916     AAX_eProperty_MaxCap = 10000 // Maximum possible property value over the lifetime of AAX
00917 }; AAX_ENUM_SIZE_CHECK(AAX_EProperty);
00918
00920 #endif // AAX_PROPERTIES_H

```

15.268 AAX_Push2ByteStructAlignment.h File Reference

15.268.1 Description

Set the struct alignment to 2-byte. This file will throw an error on platforms that do not support 2-byte alignment (i.e. TI DSPs)

When setting the alignment for a struct in order to match a particular environment (e.g. host/plugin binary compatibility) the following macros are recommended:

- [AAX_ALIGN_FILE_HOST](#)
- [AAX_ALIGN_FILE_ALG](#)
- [AAX_ALIGN_FILE_RESET](#)

15.268.2 Usage notes

- Always follow an inclusion of this file with a matching inclusion of [AAX_PopStructAlignment.h](#)

- Do not place other file `#include` after this file. For example:

```
// HeaderFile1.h
#include AAX_Push2ByteStructAlignment.h
#include HeaderFile2.h // this file now has 2-byte alignment also!!
// HeaderFile1.h definitions...
#include AAX_PopStructAlignment.h
// end HeaderFile1.h
```

This will cause problems if HeaderFile2.h is included elsewhere without the 2-byte alignment which will manifest as hard to find run-time bugs. The proper usage is:

```
// HeaderFile1.h
#include HeaderFile2.h
#include AAX_Push2ByteStructAlignment.h
// HeaderFile1.h definitions...
#include AAX_PopStructAlignment.h
// end HeaderFile1.h
```

See also

[AAX_Push4ByteStructAlignment.h](#)

[AAX_Push8ByteStructAlignment.h](#)

[AAX_PopStructAlignment.h](#)

15.269 AAX_Push2ByteStructAlignment.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00062 /*=====*/
00063
```

```

00064 #ifdef _TMS320C6X
00065 #error "TI structure packing changes not supported"
00066 #elif defined (_MSC_VER)
00067 #pragma warning( disable : 4103 ) // used #pragma pack to change alignment
00068 #pragma pack(push, 2)
00069 #elif defined (__GNUC__)
00070 // Uncomment this warning suppression if you really want to apply packing to a virtual data
00071 // structure, but note that there is no guarantee of cross-platform compatibility for such
00072 // a structure. For more information, see the AAX_ALIGN_FILE_ALG macro documentation
00073 // #ifdef __clang__
00074 //     #pragma clang diagnostic push
00075 //     #pragma clang diagnostic ignored "-Wno-incompatible-ms-struct"
00076 // #endif
00077 #pragma ms_struct on
00078 // #ifdef __clang__
00079 //     #pragma clang diagnostic pop
00080 // #endif
00081 #pragma pack(push, 2)
00082 #elif defined (__MWERKS__)
00083 #pragma options align=mac68k
00084 #else
00085 #error "You need to supply a pragma here to set structure alignment to 2 bytes"
00086 #endif
00087
00088 // Nesting of struct alignment headers is not allowed
00089 #ifdef __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00090 #error "Nested AAX struct alignment directives"
00091 #else
00092 #define __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00093 #endif

```

15.270 AAX_Push4ByteStructAlignment.h File Reference

15.270.1 Description

Set the struct alignment to 4-byte.

When setting the alignment for a struct in order to match a particular environment (e.g. host/plug-in binary compatibility) the following macros are recommended:

- [AAX_ALIGN_FILE_HOST](#)
- [AAX_ALIGN_FILE_ALG](#)
- [AAX_ALIGN_FILE_RESET](#)

15.270.2 Usage notes

- Always follow an inclusion of this file with a matching inclusion of [AAX_PopStructAlignment.h](#)
- Do not place other file `#include` after this file. For example:

```

// HeaderFile1.h
#include AAX_Push4ByteStructAlignment.h
#include HeaderFile2.h // this file now has 4-byte alignment also!!
// HeaderFile1.h definitions...
#include AAX_PopStructAlignment.h
// end HeaderFile1.h

```

This will cause problems if HeaderFile2.h is included elsewhere without the 4-byte alignment which will manifest as hard to find run-time bugs. The proper usage is:

```

// HeaderFile1.h
#include HeaderFile2.h
#include AAX_Push4ByteStructAlignment.h
// HeaderFile1.h definitions...
#include AAX_PopStructAlignment.h
// end HeaderFile1.h

```

See also

- [AAX_Push2ByteStructAlignment.h](#)
- [AAX_Push8ByteStructAlignment.h](#)
- [AAX_PopStructAlignment.h](#)

15.271 AAX_Push4ByteStructAlignment.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00061 /*=====*/
00062
00063 #ifdef _TMS320C6X
00064 // TI is OK - 4 byte alignment is the only allowed alignment
00065 #elif defined (_MSC_VER)
00066 #pragma warning( disable : 4103 ) // used #pragma pack to change alignment
00067 #pragma pack(push, 4)
00068 #elif defined (__GNUC__)
00069 // Uncomment this warning suppression if you really want to apply packing to a virtual data
00070 // structure, but note that there is no guarantee of cross-platform compatibility for such
00071 // a structure. For more information, see the AAX_ALIGN_FILE_ALG macro documentation
00072 // #ifdef __clang__
00073 //     #pragma clang diagnostic push
00074 //     #pragma clang diagnostic ignored "-Wno-incompatible-ms-struct"
00075 // #endif
00076 #pragma ms_struct on
00077 // #ifdef __clang__
00078 //     #pragma clang diagnostic pop
00079 // #endif
00080 #pragma pack(push, 4)
00081 #elif defined (__MWERKS__)
00082 #pragma options align=mac68k
00083 #else
00084 #error "You need to supply a pragma here to set structure alignment to 4 bytes"
00085 #endif
00086
00087 // Nesting of struct alignment headers is not allowed
00088 #ifdef __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00089 #error "Nested AAX struct alignment directives"
00090 #else
00091 #define __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00092 #endif

```

15.272 AAX_Push8ByteStructAlignment.h File Reference

15.272.1 Description

Set the struct alignment to 8-byte.

When setting the alignment for a struct in order to match a particular environment (e.g. host/plugin binary compatibility) the following macros are recommended:

- [AAX_ALIGN_FILE_HOST](#)
- [AAX_ALIGN_FILE_ALG](#)
- [AAX_ALIGN_FILE_RESET](#)

15.272.2 Usage notes

- Always follow an inclusion of this file with a matching inclusion of [AAX_PopStructAlignment.h](#)

- Do not place other file `#include` after this file. For example:

```
// HeaderFile1.h
#include AAX_Push8ByteStructAlignment.h
#include HeaderFile2.h // this file now has 8-byte alignment also!!
// HeaderFile1.h definitions...
#include AAX_PopStructAlignment.h
// end HeaderFile1.h
```

This will cause problems if HeaderFile2.h is included elsewhere without the 8-byte alignment which will manifest as hard to find run-time bugs. The proper usage is:

```
// HeaderFile1.h
#include HeaderFile2.h
#include AAX_Push8ByteStructAlignment.h
// HeaderFile1.h definitions...
#include AAX_PopStructAlignment.h
// end HeaderFile1.h
```

See also

[AAX_Push2ByteStructAlignment.h](#)

[AAX_Push4ByteStructAlignment.h](#)

[AAX_PopStructAlignment.h](#)

15.273 AAX_Push8ByteStructAlignment.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2015, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00061 /*=====*/
00062
00063 #ifdef _TMS320C6X
00064 // TI is OK - 8 byte alignment is the only allowed alignment
00065 #elif defined (_MSC_VER)
00066 #pragma warning( disable : 4103 ) // used #pragma pack to change alignment
00067 #pragma pack(push, 8)
00068 #elif defined (__GNUC__)
00069 // Uncomment this warning suppression if you really want to apply packing to a virtual data
00070 // structure, but note that there is no guarantee of cross-platform compatibility for such
00071 // a structure. For more information, see the AAX_ALIGN_FILE_ALG macro documentation
00072 // #ifdef __clang__
00073 // #pragma clang diagnostic push
00074 // #pragma clang diagnostic ignored "-Wno-incompatible-ms-struct"
00075 // #endif
00076 #pragma ms_struct on
00077 // #ifdef __clang__
00078 // #pragma clang diagnostic pop
00079 // #endif
00080 #pragma pack(push, 8)
00081 #elif defined (__MWERKS__)
00082 #pragma options align=mac68k
00083 #else
00084 #error "You need to supply a pragma here to set structure alignment to 8 bytes"
00085 #endif
00086
00087 // Nesting of struct alignment headers is not allowed
00088 #ifdef __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00089 #error "Nested AAX struct alignment directives"
00090 #else
00091 #define __AAX_CUSTOM_STRUCT_ALIGN_IS_SET__
00092 #endif
```

15.274 AAX_SessionDocumentTypes.h File Reference

```
#include "AAX.h"
#include <stdint.h>
#include <AAX_ALIGN_FILE_BEGIN>
#include <AAX_ALIGN_FILE_HOST>
#include <AAX_ALIGN_FILE_END>
#include <AAX_ALIGN_FILE_RESET>
```

Classes

- struct [AAX_CTempoBreakpoint](#)

Macros

- #define [AAX_SessionDocumentTypes_H](#)

Variables

- [AAX_CONSTEXPR AAX_CTypeID kAAX_DataBufferType_TempoBreakpointArray](#) = 'AXtB'

15.274.1 Macro Definition Documentation

15.274.1.1 AAX_SessionDocumentTypes_H

```
#define AAX_SessionDocumentTypes_H
```

15.274.2 Variable Documentation

15.274.2.1 kAAX_DataBufferType_TempoBreakpointArray

```
AAX\_CONSTEXPR AAX\_CTypeID kAAX\_DataBufferType\_TempoBreakpointArray = 'AXtB'
```


15.275 AAX_SessionDocumentTypes.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012  /*=====*/
00013  /*=====*/
00014  #pragma once
00015  #ifndef AAX_SessionDocumentTypes_H
00016  #define AAX_SessionDocumentTypes_H
00017
00018  #include "AAX.h"
00019  #include <stdint.h>
00020
00021  AAX_CONSTEXPR AAX_CTypeID kAAX_DataBufferType_TempoBreakpointArray = 'AXtB';
00022
00023  #include AAX_ALIGN_FILE_BEGIN
00024  #include AAX_ALIGN_FILE_HOST
00025  #include AAX_ALIGN_FILE_END
00026
00027  struct AAX_CTempoBreakpoint
00028  {
00029      int64_t mSampleLocation{0};
00030      float mValue{0.f};
00031  };
00032  static_assert(16 == sizeof(AAX_CTempoBreakpoint), "Unexpected size for AAX_CTempoBreakpoint");
00033
00034  #include AAX_ALIGN_FILE_BEGIN
00035  #include AAX_ALIGN_FILE_RESET
00036  #include AAX_ALIGN_FILE_END
00037
00038  #endif // AAX_SessionDocumentTypes_H

```

15.276 AAX_SliderConversions.h File Reference

```

#include "AAX.h"
#include <algorithm>
#include <stdint.h>

```

15.276.1 Description

Legacy utilities for converting parameter values to and from the normalized full-scale 32-bit fixed domain that was used for RTAS/TDM plug-ins.

Legacy Porting Notes These utilities may be required in order to maintain settings chunk compatibility with plug-ins that were ported from the legacy RTAS/TDM format.

Note

AAX does not provide facilities for converting to and from extended80 data types. If you use these types in your plug-in settings then you must provide your own chunk data parsing routines.

Macros

- #define [AAX_SLIDERCONVERSIONS_H](#)
- #define [AAX_LIMIT](#)(v1, firstVal, secondVal) ((secondVal > firstVal) ? (std::max)((std::min)(v1,secondVal),firstVal) : (std::min)((std::max)(v1,secondVal),firstVal))

Functions

- int32_t [LongControlToNewRange](#) (int32_t aValue, int32_t rangeMin, int32_t rangeMax)
- int32_t [LongToLongControl](#) (int32_t aValue, int32_t rangeMin, int32_t rangeMax)
Convert from int32_t control value 0x80000000...0x7FFFFFFF to a int32_t ranging from rangeMin to rangeMax (linear)
- double [LongControlToDouble](#) (int32_t aValue, double firstVal, double secondVal)
Convert from int32_t control value 0x80000000...0x7FFFFFFF to an double ranging from firstVal to secondVal (linear)
- int32_t [DoubleToLongControl](#) (double aValue, double firstVal, double secondVal)
Convert from an double ranging from firstVal to secondVal (linear) to int32_t control value 0x80000000...0x7FFFFFFF
- int32_t [DoubleToLongControlNonlinear](#) (double aValue, double *minVal, double *rangePercent, int32_t numRanges)
- double [LongControlToDoubleNonlinear](#) (int32_t aValue, double *minVal, double *rangePercent, int32_t numRanges)
- double [LongControlToLogDouble](#) (int32_t aValue, double minVal, double maxVal)
Convert from int32_t control value 0x80000000...0x7FFFFFFF to an double ranging from minVal to maxVal (logarithmic)
- int32_t [LogDoubleToLongControl](#) (double aValue, double minVal, double maxVal)
Convert from an double ranging from minVal to maxVal (logarithmic) to int32_t control value 0x80000000...0x7FFFFFFF

15.276.2 Macro Definition Documentation

15.276.2.1 AAX_SLIDERCONVERSIONS_H

```
#define AAX_SLIDERCONVERSIONS_H
```

15.276.2.2 AAX_LIMIT

```
#define AAX_LIMIT(  
    v1,  
    firstVal,  
    secondVal ) ( (secondVal > firstVal) ? (std::max)((std::min)(v1,secondVal),firstVal)  
Val) : (std::min)((std::max)(v1,secondVal),firstVal) )
```

15.276.3 Function Documentation

15.276.3.1 LongControlToNewRange()

```
int32_t LongControlToNewRange (
    int32_t aValue,
    int32_t rangeMin,
    int32_t rangeMax )
```

15.276.3.2 LongToLongControl()

```
int32_t LongToLongControl (
    int32_t aValue,
    int32_t rangeMin,
    int32_t rangeMax )
```

Convert from int32_t control value 0x80000000...0x7FFFFFFF to a int32_t ranging from rangeMin to rangeMax (linear)

15.276.3.3 LongControlToDouble()

```
double LongControlToDouble (
    int32_t aValue,
    double firstVal,
    double secondVal )
```

Convert from int32_t control value 0x80000000...0x7FFFFFFF to an double ranging from firstVal to secondVal (linear)

15.276.3.4 DoubleToLongControl()

```
int32_t DoubleToLongControl (
    double aValue,
    double firstVal,
    double secondVal )
```

Convert from an double ranging from firstVal to secondVal (linear) to int32_t control value 0x80000000...0x7FFFFFFF.

15.276.3.5 DoubleToLongControlNonlinear()

```
int32_t DoubleToLongControlNonlinear (
    double aValue,
    double * minVal,
    double * rangePercent,
    int32_t numRanges )
```

15.276.3.6 LongControlToDoubleNonlinear()

```
double LongControlToDoubleNonlinear (
    int32_t aValue,
    double * minVal,
    double * rangePercent,
    int32_t numRanges )
```

15.276.3.7 LongControlToLogDouble()

```
double LongControlToLogDouble (
    int32_t aValue,
    double minVal,
    double maxVal )
```

Convert from int32_t control value 0x80000000...0x7FFFFFFF to an double ranging from minVal to maxVal (logarithmic)

Note

This is LOGARITHMIC, so minVal & maxVal have to be > zero!

15.276.3.8 LogDoubleToLongControl()

```
int32_t LogDoubleToLongControl (
    double aValue,
    double minVal,
    double maxVal )
```

Convert from an double ranging from minVal to maxVal (logarithmic) to int32_t control value 0x80000000...0x7←FFFFFFF.

Note

This is LOGARITHMIC, so minVal & maxVal have to be > zero!

15.277 AAX_SliderConversions.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2015, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011  */
00012
00028  /*=====*/
00029
00030
00031  #pragma once
00032
00033  #ifndef AAX_SLIDERCONVERSIONS_H
00034  #define AAX_SLIDERCONVERSIONS_H
00035
00036  #include "AAX.h"
00037  #include <algorithm>
00038  #include <stdint.h>
00039
00040
00041  #define AAX_LIMIT(v1,firstVal,secondVal) ( (secondVal > firstVal) ?
    (std::max)((std::min)(v1,secondVal),firstVal) : (std::min)((std::max)(v1,secondVal),firstVal) )
00042
00043  int32_t LongControlToNewRange (int32_t aValue, int32_t rangeMin, int32_t rangeMax);
00044
00048  int32_t LongToLongControl (int32_t aValue, int32_t rangeMin, int32_t rangeMax);
00049
00053  double LongControlToDouble(int32_t aValue, double firstVal, double secondVal);
00054
00058  int32_t DoubleToLongControl (double aValue, double firstVal, double secondVal);
00059
00060  int32_t DoubleToLongControlNonlinear(double aValue, double* minVal, double* rangePercent, int32_t
    numRanges);
00061  double LongControlToDoubleNonlinear(int32_t aValue, double* minVal, double* rangePercent, int32_t
    numRanges);
00062
00069  double LongControlToLogDouble(int32_t aValue, double minVal, double maxVal);
00070
00077  int32_t LogDoubleToLongControl(double aValue, double minVal, double maxVal);
00078
00079  #endif // AAX_SLIDERCONVERSIONS_H
00080

```

15.278 AAX_StringUtilities.h File Reference

```

#include "AAX.h"
#include "AAX_Enums.h"
#include <string>
#include "AAX_StringUtilities.hpp"

```

15.278.1 Description

Various string utility definitions for AAX Native.

Namespaces

- namespace [AAX](#)

Macros

- `#define AAXLibrary_AAX_StringUtilities_h`

Functions

- `void AAX::GetCStringOfLength (char *stringOut, const char *stringIn, int32_t aMaxChars)`
=====
- `int32_t AAX::Caseless_strcmp (const char *cs, const char *ct)`
- `std::string AAX::Binary2String (uint32_t binaryValue, int32_t numBits)`
- `uint32_t AAX::String2Binary (const AAX_IString &s)`
- `bool AAX::IsASCII (char inChar)`
- `bool AAX::IsFourCharASCII (uint32_t inFourChar)`
- `std::string AAX::AsStringFourChar (uint32_t inFourChar)`
- `std::string AAX::AsStringPropertyValue (AAX_EProperty inProperty, AAX_CPropertyValue inPropertyValue)`
- `std::string AAX::AsStringInt32 (int32_t inInt32)`
- `std::string AAX::AsStringUInt32 (uint32_t inUInt32)`
- `std::string AAX::AsStringIDTriad (const AAX_SPlugInIdentifierTriad &inIDTriad)`
- `std::string AAX::AsStringStemFormat (AAX_EStemFormat inStemFormat, bool inAbbreviate=false)`
- `std::string AAX::AsStringStemChannel (AAX_EStemFormat inStemFormat, uint32_t inChannelIndex, bool inAbbreviate)`
- `std::string AAX::AsStringResult (AAX_Result inResult)`

15.278.2 Macro Definition Documentation

15.278.2.1 AAXLibrary_AAX_StringUtilities_h

```
#define AAXLibrary_AAX_StringUtilities_h
```

15.279 AAX_StringUtilities.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2016, 2018, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  */
00011
00017 /*=====*/
00018 #pragma once
00019
00020 #ifndef AAXLibrary_AAX_StringUtilities_h
00021 #define AAXLibrary_AAX_StringUtilities_h
00022
00023 // AAX headers
00024 #include "AAX.h"
00025 #include "AAX_Enums.h"
00026
00027 // Standard Library headers
00028 #include <string>
```

```

00029
00030 class AAX_IString;
00031
00032
00033 //-----
00034 #pragma mark Utility functions
00035
00036 namespace AAX
00037 {
00038     inline void          GetCStringOfLength(char *stringOut, const char* stringIn, int32_t aMaxChars);
00039     inline int32_t        Caseless_strcmp(const char* cs, const char* ct);
00040
00041     inline std::string    Binary2String(uint32_t binaryValue, int32_t numBits);
00042     inline uint32_t        String2Binary(const AAX_IString& s);
00043
00044     inline bool          IsASCII(char inChar);
00045     inline bool          IsFourCharASCII(uint32_t inFourChar);
00046
00047     inline std::string    AsStringFourChar(uint32_t inFourChar);
00048     inline std::string    AsStringPropertyValue(AAX_EProperty inProperty, AAX_CPropertyValue
inPropertyValue);
00049     inline std::string    AsStringInt32(int32_t inInt32);
00050     inline std::string    AsStringUInt32(uint32_t inUInt32);
00051     inline std::string    AsStringIDTriad(const AAX_SPlugInIdentifierTriad& inIDTriad);
00052     inline std::string    AsStringStemFormat(AAX_EStemFormat inStemFormat, bool inAbbreviate = false);
00053     inline std::string    AsStringStemChannel(AAX_EStemFormat inStemFormat, uint32_t inChannelIndex,
bool inAbbreviate);
00054     inline std::string    AsStringResult(AAX_Result inResult);
00055 } // namespace AAX
00056
00057
00058 //-----
00059 #pragma mark Implementation header
00060
00061 #include "AAX_StringUtilities.hpp"
00062
00063
00064 #endif /* AAXLibrary_AAX_StringUtilities_h */

```

15.280 AAX_StringUtilities.hpp File Reference

```

#include "AAX_IString.h"
#include "AAX_Errors.h"
#include "AAX_Assert.h"
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <sstream>
#include <string>
#include <vector>

```

Namespaces

- namespace [AAX](#)
- namespace [AAX::internal](#)

Macros

- #define [DEFINE_AAX_ERROR_STRING](#)(X) if (X == inResult) { return std::string(#X); }

Functions

- template<typename T >
std::string [AAX::internal::ToHexadecimal](#) (T inValue, bool inLeadingZeros=false)

15.280.1 Macro Definition Documentation

15.280.1.1 DEFINE_AAX_ERROR_STRING

```
#define DEFINE_AAX_ERROR_STRING(
    X ) if (X == inResult) { return std::string(#X); }
```

15.281 AAX_StringUtilities.hpp

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2014-2017, 2019-2021, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010 /*=====*/
00011
00012 #ifndef AAXLibrary_AAX_StringUtilities_hpp
00013 #define AAXLibrary_AAX_StringUtilities_hpp
00014
00015 #include "AAX_IString.h"
00016 #include "AAX_Errors.h"
00017 #include "AAX_Assert.h"
00018
00019 #include <cstdlib>
00020 #include <cstring>
00021
00022 #include <algorithm>
00023 #include <sstream>
00024 #include <string>
00025 #include <vector>
00026
00027 //=====
00028 //
00029 // FloatToString: Convert the given floating point number to a pascal string.
00030 //
00031 //=====
00032 /*
00033 void FloatToString(float aNumber, StringPtr aString)
00034 {
00035     Str255 MantissaStr;
00036     double aDouble;
00037     StringPtr tempStr;
00038     int32_t mantissa,tens,hundreds;
00039     int16_t count;
00040
00041     aDouble = (double) aNumber;
00042     if (aNumber < 0.0) // take abs value
00043         aDouble = -aDouble;
00044
00045     aDouble += 0.005; // perform rounding by adding 1/2 of the hundreths digit
00046
00047     mantissa = aDouble;
00048     tens = (aDouble * 10.0) - (mantissa * 10.0);
00049     hundreds = (aDouble * 100.0) - (mantissa * 100.0) - (tens * 10.0);
00050
00051     NumToString(mantissa, MantissaStr);
00052
00053     // set up string length
00054     if (aNumber < 0.0)
00055         *aString++ = (char) (1 + 3 + *MantissaStr);
00056     else
00057         *aString++ = (char) (3 + *MantissaStr);
00058
00059     tempStr = MantissaStr;
00060
00061     // copy mantissa first
00062     count = *tempStr++;
```



```

00063
00064     if (aNumber < 0.0)
00065         *aString++ = '-';
00066
00067     while (count--)
00068         *aString++ = *tempStr++;
00069
00070     *aString++ = '.';
00071     *aString++ = (char) (tens + '0');
00072     *aString++ = (char) (hundreds + '0');
00073 }
00074 */
00075
00077 //
00078 //  GetCStringOfLength
00079 //
00080 //  A routine for selecting a string based on the size passed in
00081 //  by the client.  If none of the strings are short enough then
00082 //  the shortest string is truncated to fit.
00083 //
00084 //  stringIn="A Very Nice String\nA String\nAString\nStr\n";
00085 //
00086 //  Submitted from Erik Gavriluk of BombFactory (Free of Charge)
00087 //  Debugged and separator character changed by Frederick Umminger
00088 //=====
00089
00090 void AAX::GetCStringOfLength(char *s_out, const char* s_in, int32_t aMaxChars)
00091 {
00092     AAX_ASSERT(0 < aMaxChars);
00093
00094     const char kSeparator = '\n';
00095
00096     if(s_in)
00097     {
00098         const char* s_begin = s_in;
00099         const char* s_end = s_begin;
00100         while(s_begin)
00101         {
00102             // Count characters in current substring
00103             while((*s_end != kSeparator) && (*s_end != '\0'))
00104             {
00105                 s_end++;
00106             };
00107
00108             // If substring is less than or equal to aMaxChars then use it.
00109             if((s_end-s_begin <= aMaxChars) || (*s_end=='\0'))
00110             {
00111                 break;
00112             }
00113
00114             s_begin = ++s_end;
00115         }
00116         // We don't use strncpy in order to make sure a '\0' gets put on the end of s_out
00117         *s_out = '\0';
00118         const int32_t length = int32_t(s_end-s_begin);
00119         if (0 < length && 0 < aMaxChars)
00120         {
00121             std::strncat(s_out, s_begin, static_cast<std::size_t>(std::max<int32_t>(0,
00122 std::min<int32_t>(aMaxChars,length))));
00123         }
00124     }
00125     else if (0 < aMaxChars)
00126     {
00127         strncpy(s_out, "", static_cast<size_t>(aMaxChars));
00128     };
00129 }
00130 int32_t AAX::Caseless_strcmp(const char* cs, const char* ct)
00131 {
00132     if(cs)
00133     {
00134         if(ct)
00135         {
00136             while(*cs && *ct)
00137             {
00138                 int32_t cmp = toupper(*ct++) - toupper(*cs++);
00139                 if(cmp) return cmp;
00140             };
00141             if(*cs)
00142             {
00143                 return -1;
00144             }
00145             else
00146             {
00147                 if(*ct)
00148                 {
00149                     return 1;

```

```

00150         }
00151         else
00152         {
00153             return 0;
00154         };
00155     };
00156 }
00157 else
00158 {
00159     return -1;
00160 };
00161 }
00162 else
00163 {
00164     if(ct)
00165         return 1;
00166     else
00167         return 0;
00168 }
00169 }
00170 }
00171
00172
00173 std::string AAX::Binary2String(uint32_t value, int32_t numBits)
00174 {
00175     std::string s;
00176
00177     uint32_t currentBitMask = (static_cast<uint32_t>(0x1) << (numBits-1));
00178
00179     while (currentBitMask != 0)
00180     {
00181         if (currentBitMask & value)
00182         {
00183             s += "1";
00184         }
00185         else
00186         {
00187             s += "0";
00188         };
00189         currentBitMask >>= 1;
00190     }
00191     return s;
00192 }
00193
00194 uint32_t AAX::String2Binary(const AAX_IString& s)
00195 {
00196     uint32_t value = 0;
00197
00198     const char* const cS = s.Get();
00199     int32_t length = int32_t(s.Length());
00200     for(int32_t i = 0; i < length ; i++)
00201     {
00202         switch(cS[i])
00203         {
00204             case '0':
00205                 break;
00206             case '1':
00207                 value |= (0x1 << (length-1-i));
00208                 break;
00209             default:
00210                 AAX_ASSERT('0' == cS[i] || '1' == cS[i]);
00211         };
00212     };
00213
00214     return value;
00215 }
00216
00217 bool AAX::IsASCII(char inChar)
00218 {
00219     return (0x20 <= inChar) && (0x7E >= inChar);
00220 }
00221
00222 bool AAX::IsFourCharASCII(uint32_t inFourChar)
00223 {
00224     const uint32_t oneCharMask = 0x000000FF;
00225     const size_t oneCharNumBits = 8;
00226
00227     bool result = true;
00228     for (uint16_t i = 3; true == result /* i value checked within loop */; --i)
00229     {
00230         const char curChar = static_cast<const char>((inFourChar >> (i*oneCharNumBits)) & oneCharMask);
00231         result = result && IsASCII(curChar);
00232         if (0 == i) { break; }
00233     }
00234     return result;
00235 }
00236

```

```

00237 std::string AAX::AsStringFourChar(uint32_t inFourChar)
00238 {
00239     AAX_CONSTEXPR uint32_t oneCharMask = 0x000000FF;
00240     AAX_CONSTEXPR int16_t oneCharNumBits = 8;
00241     AAX_CONSTEXPR auto unknownChar = "(?)"; // for current usage, a raw string here is slightly more
        efficient than a std::string
00242
00243     std::string resultStr;
00244     for (int16_t i = 3; i >= 0; --i)
00245     {
00246         const char curChar = static_cast<char>((inFourChar >> (i*oneCharNumBits)) & oneCharMask);
00247
00248         // Prefer an explicit 'if' statement instead of a ternary operator to allow using the most
00249         // efficient 'append' operator in each case
00250         if (IsASCII(curChar))
00251         {
00252             resultStr += curChar;
00253         }
00254         else
00255         {
00256             resultStr += unknownChar;
00257         }
00258     }
00259     return resultStr;
00260 }
00261
00262 namespace AAX { namespace internal {
00263 template <typename T>
00264 std::string ToHexadecimal(T inValue, bool inLeadingZeros = false)
00265 {
00266     AAX_CONSTEXPR char hexChars[] = "0123456789abcdef";
00267     AAX_CONSTEXPR size_t size = sizeof(T) * 2;
00268
00269     std::string buffer{"0"};
00270
00271     // This conditional is to respect the expected output on 'inValud=0': "0" (instead of "0x0")
00272     if (inValue)
00273     {
00274         buffer += 'x';
00275         bool first_non_zero = inLeadingZeros;
00276
00277         // Largest integers will have 16 hex characters, just below the short-string
00278         // optimization of std::string, so no dynamic allocation is required
00279         for (size_t i = 0; i < size; ++i)
00280         {
00281             const auto c = hexChars[(inValue >> 4 * (size - 1 - i)) & 0xf];
00282             if (first_non_zero || c != '0')
00283             {
00284                 first_non_zero = true;
00285                 buffer += c;
00286             }
00287         }
00288     }
00289
00290     return buffer;
00291 }
00292 }}
00293
00294 std::string AAX::AsStringPropertyValue(AAX_EProperty inProperty, AAX_CPropertyValue inPropertyValue)
00295 {
00296     // Attempt to infer a sensible way to print the property
00297     if (AAX_eProperty_SampleRate == inProperty ||
00298         AAX_eProperty_Constraint_Location == inProperty)
00299     {
00300         // Print specific properties' values as bitfield
00301
00302         // We want the exact bits, so we memcpy to avoid any potential issues
00303         // with casting from signed to unsigned
00304         uint32_t bitfield;
00305         memcpy(&bitfield, &inPropertyValue, sizeof(uint32_t));
00306
00307         AAX_CONSTEXPR int32_t maxNumBitsToShow = 8; // Currently there are no bitfield properties with
        more than 8 possible flags
00308         return AAX::Binary2String(bitfield, maxNumBitsToShow);
00309     }
00310
00311     if (AAX::IsFourCharASCII(static_cast<uint32_t>(inPropertyValue)))
00312     {
00313         // Print values in ASCII range as four-char
00314         return '\\' + AAX::AsStringFourChar(static_cast<uint32_t>(inPropertyValue)) + '\\';
00315     }
00316
00317     if (0x00FFFFFF < abs(inPropertyValue))
00318     {
00319         // Print values with most bits used as hex
00320         return internal::ToHexadecimal(inPropertyValue);
00321     }

```

```

00322
00323 // Otherwise, print as simple decimal
00324 return std::to_string(static_cast<long int>(inPropertyValue));
00325 }
00326
00327 std::string AAX::AsStringInt32(int32_t inInt32)
00328 {
00329     return std::to_string((long int)inInt32);
00330 }
00331
00332 std::string AAX::AsStringUInt32(uint32_t inUInt32)
00333 {
00334     return std::to_string((unsigned long)inUInt32);
00335 }
00336
00337 std::string AAX::AsStringIDTriad(const AAX_SPlugInIdentifierTriad& inIDTriad)
00338 {
00339     std::string result = "(";
00340
00341     result += "man: ' " + AAX::AsStringFourChar(inIDTriad.mManufacturerID) + "', ";
00342     result += "prod: ' " + AAX::AsStringFourChar(inIDTriad.mProductID) + "', ";
00343     result += "type: ' " + AAX::AsStringFourChar(inIDTriad.mPlugInID) + "'";
00344
00345     result += ")";
00346     return result;
00347 }
00348
00349 std::string AAX::AsStringStemFormat(AAX_EStemFormat inStemFormat, bool inAbbreviate)
00350 {
00351     switch (inStemFormat)
00352     {
00353         case AAX_eStemFormat_Mono: { return std::string("Mono"); break; }
00354         case AAX_eStemFormat_Stereo: { return std::string(inAbbreviate ? "St" : "Stereo"); break; }
00355         case AAX_eStemFormat_LCR: { return std::string("LCR"); break; }
00356         case AAX_eStemFormat_LCRS: { return std::string("LCRS"); break; }
00357         case AAX_eStemFormat_Quad: { return std::string("Quad"); break; }
00358         case AAX_eStemFormat_5_0: { return std::string("5.0"); break; }
00359         case AAX_eStemFormat_5_1: { return std::string("5.1"); break; }
00360         case AAX_eStemFormat_6_0: { return std::string("6.0"); break; }
00361         case AAX_eStemFormat_6_1: { return std::string("6.1"); break; }
00362         case AAX_eStemFormat_7_0_SDDS: { return std::string(inAbbreviate ? "7.0 S" : "7.0 SDDS");
00363         break; }
00364         case AAX_eStemFormat_7_1_SDDS: { return std::string(inAbbreviate ? "7.1 S" : "7.1 SDDS");
00365         break; }
00366         case AAX_eStemFormat_7_0_DTS: { return std::string("7.0"); break; }
00367         case AAX_eStemFormat_7_1_DTS: { return std::string("7.1"); break; }
00368         case AAX_eStemFormat_7_0_2: { return std::string("7.0.2"); break; }
00369         case AAX_eStemFormat_7_1_2: { return std::string("7.1.2"); break; }
00370         case AAX_eStemFormat_Ambi_1_ACN: { return std::string(inAbbreviate ? "Amb1" : "Ambisonics (1st
00371         Order)"); break; }
00372         case AAX_eStemFormat_Ambi_2_ACN: { return std::string(inAbbreviate ? "Amb2" : "Ambisonics (2nd
00373         Order)"); break; }
00374         case AAX_eStemFormat_Ambi_3_ACN: { return std::string(inAbbreviate ? "Amb3" : "Ambisonics (3rd
00375         Order)"); break; }
00376         case AAX_eStemFormat_Ambi_4_ACN: { return std::string(inAbbreviate ? "Amb4" : "Ambisonics (4th
00377         Order)"); break; }
00378         case AAX_eStemFormat_Ambi_5_ACN: { return std::string(inAbbreviate ? "Amb5" : "Ambisonics (5th
00379         Order)"); break; }
00380         case AAX_eStemFormat_Ambi_6_ACN: { return std::string(inAbbreviate ? "Amb6" : "Ambisonics (6th
00381         Order)"); break; }
00382         case AAX_eStemFormat_Ambi_7_ACN: { return std::string(inAbbreviate ? "Amb7" : "Ambisonics (7th
00383         Order)"); break; }
00384         case AAX_eStemFormat_5_0_2: { return std::string("5.0.2"); break; }
00385         case AAX_eStemFormat_5_1_2: { return std::string("5.1.2"); break; }
00386         case AAX_eStemFormat_5_0_4: { return std::string("5.0.4"); break; }
00387         case AAX_eStemFormat_5_1_4: { return std::string("5.1.4"); break; }
00388         case AAX_eStemFormat_7_0_4: { return std::string("7.0.4"); break; }
00389         case AAX_eStemFormat_7_1_4: { return std::string("7.1.4"); break; }
00390         case AAX_eStemFormat_7_0_6: { return std::string("7.0.6"); break; }
00391         case AAX_eStemFormat_7_1_6: { return std::string("7.1.6"); break; }
00392         case AAX_eStemFormat_9_0_4: { return std::string("9.0.4"); break; }
00393         case AAX_eStemFormat_9_1_4: { return std::string("9.1.4"); break; }
00394         case AAX_eStemFormat_9_0_6: { return std::string("9.0.6"); break; }
00395         case AAX_eStemFormat_9_1_6: { return std::string("9.1.6"); break; }
00396
00397         case AAX_eStemFormat_None: { return std::string("None"); break; }
00398         case AAX_eStemFormat_Any: { return std::string("Any"); break; }
00399
00400         case AAX_eStemFormat_INT32_MAX:
00401         case AAX_eStemFormatNum:
00402         default: { return std::string(inAbbreviate ? "unk" : "unknown stem format"); break; }
00403     }
00404 }
00405
00406 std::string AAX::AsStringStemChannel(AAX_EStemFormat inStemFormat, uint32_t inChannelIndex, bool
inAbbreviate)

```

```

00399 {
00400     switch (inStemFormat)
00401     {
00402         case AAX_eStemFormat_Mono:
00403             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "M" : "Audio"); }
00404             break;
00405         case AAX_eStemFormat_Stereo:
00406             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00407             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00408             break;
00409         case AAX_eStemFormat_LCR:
00410             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00411             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00412             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00413             break;
00414         case AAX_eStemFormat_LCRS:
00415             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00416             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00417             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00418             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "S" : "Surround"); }
00419             break;
00420         case AAX_eStemFormat_Quad:
00421             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00422             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00423             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00424             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00425             break;
00426         case AAX_eStemFormat_5_0:
00427             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00428             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00429             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00430             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00431             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00432             break;
00433         case AAX_eStemFormat_5_1:
00434             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00435             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00436             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00437             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00438             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00439             if (0 == inChannelIndex--) { return std::string("LFE"); }
00440             break;
00441         case AAX_eStemFormat_6_0:
00442             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00443             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00444             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00445             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00446             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Cs" : "Center Surround"); }
00447             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00448             break;
00449         case AAX_eStemFormat_6_1:
00450             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00451             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00452             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00453             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00454             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Cs" : "Center Surround"); }
00455             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00456             if (0 == inChannelIndex--) { return std::string("LFE"); }
00457             break;
00458         case AAX_eStemFormat_7_0_SDDS:
00459             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00460             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lc" : "Left Center"); }
00461             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00462             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rc" : "Right Center"); }
00463             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00464             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00465             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00466             break;
00467         case AAX_eStemFormat_7_1_SDDS:
00468             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00469             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lc" : "Left Center"); }
00470             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00471             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rc" : "Right Center"); }
00472             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00473             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00474             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00475             if (0 == inChannelIndex--) { return std::string("LFE"); }
00476             break;
00477         case AAX_eStemFormat_7_0_DTS:
00478             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00479             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00480             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00481             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00482             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround

```

```

        Side"); }
00483         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00484         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00485         break;
00486         case AAX_eStemFormat_7_1_DTS:
00487             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00488             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00489             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00490             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00491             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00492             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00493             if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00494             if (0 == inChannelIndex--) { return std::string("LFE"); }
00495             break;
00496             case AAX_eStemFormat_7_0_2:
00497                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00498                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00499                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00500                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00501                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00502                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00503                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00504                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "LTS" : "Left Top
Surround"); }
00505                 if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "RTS" : "Right Top
Surround"); }
00506                 break;
00507                 case AAX_eStemFormat_7_1_2:
00508                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00509                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00510                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00511                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00512                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00513                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00514                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00515                     if (0 == inChannelIndex--) { return std::string("LFE"); }
00516                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "LTS" : "Left Top
Surround"); }
00517                     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "RTS" : "Right Top
Surround"); }
00518                     break;
00519                     case AAX_eStemFormat_Ambi_1_ACN:
00520                     case AAX_eStemFormat_Ambi_2_ACN:
00521                     case AAX_eStemFormat_Ambi_3_ACN:
00522                     case AAX_eStemFormat_Ambi_4_ACN:
00523                     case AAX_eStemFormat_Ambi_5_ACN:
00524                     case AAX_eStemFormat_Ambi_6_ACN:
00525                     case AAX_eStemFormat_Ambi_7_ACN:
00526                         if (0 == inChannelIndex--) { return std::string("1"); }
00527                         if (0 == inChannelIndex--) { return std::string("2"); }
00528                         if (0 == inChannelIndex--) { return std::string("3"); }
00529                         if (0 == inChannelIndex--) { return std::string("4"); }
00530                         if (0 == inChannelIndex--) { return std::string("5"); }
00531                         if (0 == inChannelIndex--) { return std::string("6"); }
00532                         if (0 == inChannelIndex--) { return std::string("7"); }
00533                         if (0 == inChannelIndex--) { return std::string("8"); }
00534                         if (0 == inChannelIndex--) { return std::string("9"); }
00535                         if (0 == inChannelIndex--) { return std::string("10"); }
00536                         if (0 == inChannelIndex--) { return std::string("11"); }
00537                         if (0 == inChannelIndex--) { return std::string("12"); }
00538                         if (0 == inChannelIndex--) { return std::string("13"); }
00539                         if (0 == inChannelIndex--) { return std::string("14"); }
00540                         if (0 == inChannelIndex--) { return std::string("15"); }
00541                         if (0 == inChannelIndex--) { return std::string("16"); }
00542                         if (0 == inChannelIndex--) { return std::string("17"); }
00543                         if (0 == inChannelIndex--) { return std::string("18"); }
00544                         if (0 == inChannelIndex--) { return std::string("19"); }
00545                         if (0 == inChannelIndex--) { return std::string("20"); }
00546                         if (0 == inChannelIndex--) { return std::string("21"); }
00547                         if (0 == inChannelIndex--) { return std::string("22"); }
00548                         if (0 == inChannelIndex--) { return std::string("23"); }
00549                         if (0 == inChannelIndex--) { return std::string("24"); }
00550                         if (0 == inChannelIndex--) { return std::string("25"); }

```

```

00551         if (0 == inChannelIndex--) { return std::string("26"); }
00552         if (0 == inChannelIndex--) { return std::string("27"); }
00553         if (0 == inChannelIndex--) { return std::string("28"); }
00554         if (0 == inChannelIndex--) { return std::string("29"); }
00555         if (0 == inChannelIndex--) { return std::string("30"); }
00556         if (0 == inChannelIndex--) { return std::string("31"); }
00557         if (0 == inChannelIndex--) { return std::string("32"); }
00558         if (0 == inChannelIndex--) { return std::string("33"); }
00559         if (0 == inChannelIndex--) { return std::string("34"); }
00560         if (0 == inChannelIndex--) { return std::string("35"); }
00561         if (0 == inChannelIndex--) { return std::string("36"); }
00562         if (0 == inChannelIndex--) { return std::string("37"); }
00563         if (0 == inChannelIndex--) { return std::string("38"); }
00564         if (0 == inChannelIndex--) { return std::string("39"); }
00565         if (0 == inChannelIndex--) { return std::string("40"); }
00566         if (0 == inChannelIndex--) { return std::string("41"); }
00567         if (0 == inChannelIndex--) { return std::string("42"); }
00568         if (0 == inChannelIndex--) { return std::string("43"); }
00569         if (0 == inChannelIndex--) { return std::string("44"); }
00570         if (0 == inChannelIndex--) { return std::string("45"); }
00571         if (0 == inChannelIndex--) { return std::string("46"); }
00572         if (0 == inChannelIndex--) { return std::string("47"); }
00573         if (0 == inChannelIndex--) { return std::string("48"); }
00574         if (0 == inChannelIndex--) { return std::string("49"); }
00575         if (0 == inChannelIndex--) { return std::string("50"); }
00576         if (0 == inChannelIndex--) { return std::string("51"); }
00577         if (0 == inChannelIndex--) { return std::string("52"); }
00578         if (0 == inChannelIndex--) { return std::string("53"); }
00579         if (0 == inChannelIndex--) { return std::string("54"); }
00580         if (0 == inChannelIndex--) { return std::string("55"); }
00581         if (0 == inChannelIndex--) { return std::string("56"); }
00582         if (0 == inChannelIndex--) { return std::string("57"); }
00583         if (0 == inChannelIndex--) { return std::string("58"); }
00584         if (0 == inChannelIndex--) { return std::string("59"); }
00585         if (0 == inChannelIndex--) { return std::string("60"); }
00586         if (0 == inChannelIndex--) { return std::string("61"); }
00587         if (0 == inChannelIndex--) { return std::string("62"); }
00588         if (0 == inChannelIndex--) { return std::string("63"); }
00589         if (0 == inChannelIndex--) { return std::string("64"); }
00590         break;
00591     case AAX_eStemFormat_5_0_2:
00592         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00593         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00594         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00595         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00596         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00597         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltm" : "Left Top Middle"); }
00598     }
00599     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtm" : "Right Top
Middle"); }
00600     break;
00601     case AAX_eStemFormat_5_1_2:
00602         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00603         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00604         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00605         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00606         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00607         if (0 == inChannelIndex--) { return std::string("LFE"); }
00608         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltm" : "Left Top Middle"); }
00609     }
00610     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtm" : "Right Top
Middle"); }
00611     break;
00612     case AAX_eStemFormat_5_0_4:
00613         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00614         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00615         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00616         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00617         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00618         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
00619     }
00620     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
00621     }
00622     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00623     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
00624     }
00625     break;
00626     case AAX_eStemFormat_5_1_4:
00627         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00628         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00629         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00630         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ls" : "Left Surround"); }
00631         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rs" : "Right Surround"); }
00632         if (0 == inChannelIndex--) { return std::string("LFE"); }
00633         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
00634     }
00635     if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }

```



```

    }
00630         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00631         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
    }
00632     break;
00633     case AAX_eStemFormat_7_0_4:
00634         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00635         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00636         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00637         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00638         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00639         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00640         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00641         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
    }
00642         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
    }
00643         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00644         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
    }
00645     break;
00646     case AAX_eStemFormat_7_1_4:
00647         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00648         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00649         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00650         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00651         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00652         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00653         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00654         if (0 == inChannelIndex--) { return std::string("LFE"); }
00655         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
    }
00656         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
    }
00657         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00658         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
    }
00659     break;
00660     case AAX_eStemFormat_7_0_6:
00661         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00662         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00663         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00664         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00665         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00666         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00667         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00668         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
    }
00669         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
    }
00670         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltm" : "Left Top Middle"); }
    }
00671         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtm" : "Right Top
Middle"); }
00672         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00673         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
    }
00674     break;
00675     case AAX_eStemFormat_7_1_6:
00676         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00677         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00678         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00679         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00680         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00681         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00682         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00683         if (0 == inChannelIndex--) { return std::string("LFE"); }
00684         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
    }
00685         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
    }

```



```

00686         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltm" : "Left Top Middle"); }
00687     Middle"); }
00688         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00689         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
00690     }
00691     break;
00692     case AAX_eStemFormat_9_0_4:
00693         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00694         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00695         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00696         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lw" : "Left Wide"); }
00697         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rw" : "Right Wide"); }
00698         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00699         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00700         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00701         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00702         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
00703         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
00704         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00705         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
00706     }
00707     break;
00708     case AAX_eStemFormat_9_1_4:
00709         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00710         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00711         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00712         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lw" : "Left Wide"); }
00713         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rw" : "Right Wide"); }
00714         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00715         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00716         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00717         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00718         if (0 == inChannelIndex--) { return std::string("LFE"); }
00719         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
00720         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
00721         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00722         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
00723     }
00724     break;
00725     case AAX_eStemFormat_9_0_6:
00726         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00727         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00728         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00729         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lw" : "Left Wide"); }
00730         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rw" : "Right Wide"); }
00731         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }
00732         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00733         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00734         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00735         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front"); }
00736         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front"); }
00737         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltm" : "Left Top Middle"); }
00738         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtm" : "Right Top
Middle"); }
00739         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00740         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear"); }
00741     }
00742     break;
00743     case AAX_eStemFormat_9_1_6:
00744         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "L" : "Left"); }
00745         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "C" : "Center"); }
00746         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "R" : "Right"); }
00747         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lw" : "Left Wide"); }
00748         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rw" : "Right Wide"); }
00749         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lss" : "Left Surround
Side"); }

```

```

00746         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rss" : "Right Surround
Side"); }
00747         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Lsr" : "Left Surround
Rear"); }
00748         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rsr" : "Right Surround
Rear"); }
00749         if (0 == inChannelIndex--) { return std::string("LFE"); }
00750         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltf" : "Left Top Front");
}
00751         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtf" : "Right Top Front");
}
00752         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltm" : "Left Top Middle");
}
00753         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtm" : "Right Top
Middle"); }
00754         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Ltr" : "Left Top Rear"); }
00755         if (0 == inChannelIndex--) { return std::string(inAbbreviate ? "Rtr" : "Right Top Rear");
}
00756         break;
00757
00758
00759         case AAX_eStemFormat_None:
00760             break;
00761         case AAX_eStemFormat_Any:
00762             break;
00763
00764         case AAX_eStemFormat_INT32_MAX:
00765         case AAX_eStemFormatNum:
00766             default:
00767                 break;
00768     }
00769     return std::string(inAbbreviate ? "?" : "unknown");
00770 }
00771 }
00772
00773 std::string AAX::AsStringResult(AAX_Result inResult)
00774 {
00775     #ifdef DEFINE_AAX_ERROR_STRING
00776     #undef DEFINE_AAX_ERROR_STRING
00777     #endif
00778     #define DEFINE_AAX_ERROR_STRING(X) if (X == inResult) { return std::string(#X); }
00779
00780     DEFINE_AAX_ERROR_STRING(AAX_SUCCESS);
00781     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_PARAMETER_ID);
00782     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_STRING_CONVERSION);
00783     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_METER_INDEX);
00784     DEFINE_AAX_ERROR_STRING(AAX_ERROR_NULL_OBJECT);
00785     DEFINE_AAX_ERROR_STRING(AAX_ERROR_OLDER_VERSION);
00786     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_CHUNK_INDEX);
00787     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_CHUNK_ID);
00788     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INCORRECT_CHUNK_SIZE);
00789     DEFINE_AAX_ERROR_STRING(AAX_ERROR_UNIMPLEMENTED);
00790     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_PARAMETER_INDEX);
00791     DEFINE_AAX_ERROR_STRING(AAX_ERROR_NOT_INITIALIZED);
00792     DEFINE_AAX_ERROR_STRING(AAX_ERROR_ACF_ERROR);
00793     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_METER_TYPE);
00794     DEFINE_AAX_ERROR_STRING(AAX_ERROR_CONTEXT_ALREADY_HAS_METERS);
00795     DEFINE_AAX_ERROR_STRING(AAX_ERROR_NULL_COMPONENT);
00796     DEFINE_AAX_ERROR_STRING(AAX_ERROR_PORT_ID_OUT_OF_RANGE);
00797     DEFINE_AAX_ERROR_STRING(AAX_ERROR_FIELD_TYPE_DOES_NOT_SUPPORT_DIRECT_ACCESS);
00798     DEFINE_AAX_ERROR_STRING(AAX_ERROR_DIRECT_ACCESS_OUT_OF_BOUNDS);
00799     DEFINE_AAX_ERROR_STRING(AAX_ERROR_FIFO_FULL);
00800     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INITIALIZING_PACKET_STREAM_THREAD);
00801     DEFINE_AAX_ERROR_STRING(AAX_ERROR_POST_PACKET_FAILED);
00802     DEFINE_AAX_ERROR_STRING(AAX_RESULT_PACKET_STREAM_NOT_EMPTY);
00803     DEFINE_AAX_ERROR_STRING(AAX_RESULT_ADD_FIELD_UNSUPPORTED_FIELD_TYPE);
00804     DEFINE_AAX_ERROR_STRING(AAX_ERROR_MIXER_THREAD_FALLING_BEHIND);
00805     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_FIELD_INDEX);
00806     DEFINE_AAX_ERROR_STRING(AAX_ERROR_MALFORMED_CHUNK);
00807     DEFINE_AAX_ERROR_STRING(AAX_ERROR_TOD_BEHIND);
00808     DEFINE_AAX_ERROR_STRING(AAX_RESULT_NEW_PACKET_POSTED);
00809     DEFINE_AAX_ERROR_STRING(AAX_ERROR_PLUGIN_NOT_AUTHORIZED);
00810     DEFINE_AAX_ERROR_STRING(AAX_ERROR_PLUGIN_NULL_PARAMETER);
00811     DEFINE_AAX_ERROR_STRING(AAX_ERROR_NOTIFICATION_FAILED);
00812     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_VIEW_SIZE);
00813     DEFINE_AAX_ERROR_STRING(AAX_ERROR_SIGNED_INT_OVERFLOW);
00814     DEFINE_AAX_ERROR_STRING(AAX_ERROR_NO_COMPONENTS);
00815     DEFINE_AAX_ERROR_STRING(AAX_ERROR_DUPLICATE_EFFECT_ID);
00816     DEFINE_AAX_ERROR_STRING(AAX_ERROR_DUPLICATE_TYPE_ID);
00817     DEFINE_AAX_ERROR_STRING(AAX_ERROR_EMPTY_EFFECT_NAME);
00818     DEFINE_AAX_ERROR_STRING(AAX_ERROR_UNKNOWN_PLUGIN);
00819     DEFINE_AAX_ERROR_STRING(AAX_ERROR_PROPERTY_UNDEFINED);
00820     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_PATH);
00821     DEFINE_AAX_ERROR_STRING(AAX_ERROR_UNKNOWN_ID);
00822     DEFINE_AAX_ERROR_STRING(AAX_ERROR_UNKNOWN_EXCEPTION);
00823     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_ARGUMENT);
00824     DEFINE_AAX_ERROR_STRING(AAX_ERROR_NULL_ARGUMENT);

```

```

00825     DEFINE_AAX_ERROR_STRING(AAX_ERROR_INVALID_INTERNAL_DATA);
00826     DEFINE_AAX_ERROR_STRING(AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW);
00827
00828     if (AAX_ERROR_PLUGIN_BEGIN >= inResult && AAX_ERROR_PLUGIN_END <= inResult)
00829         return std::string("plug-in defined error");
00830
00831     return std::string("<unknown error code>");
00832 }
00833
00834 #endif

```

15.282 AAX_TransportTypes.h File Reference

```

#include "AAX.h"
#include <string>
#include <sstream>
#include <AAX_ALIGN_FILE_BEGIN>
#include <AAX_ALIGN_FILE_HOST>
#include <AAX_ALIGN_FILE_END>
#include <AAX_ALIGN_FILE_RESET>

```

15.282.1 Description

Structures, enums and other definitions used in transport.

Classes

- struct [AAX_TransportStateInfo_V1](#)

Functions

- bool [operator==](#) (const [AAX_TransportStateInfo_V1](#) &state1, const [AAX_TransportStateInfo_V1](#) &state2)
- bool [operator!=](#) (const [AAX_TransportStateInfo_V1](#) &state1, const [AAX_TransportStateInfo_V1](#) &state2)

15.282.2 Function Documentation

15.282.2.1 operator==()

```

bool operator== (
    const AAX\_TransportStateInfo\_V1 & state1,
    const AAX\_TransportStateInfo\_V1 & state2 ) [inline]

```

References [AAX_TransportStateInfo_V1::mIsLoopEnabled](#), [AAX_TransportStateInfo_V1::mIsRecordEnabled](#), [AAX_TransportStateInfo_V1::mIsRecording](#), [AAX_TransportStateInfo_V1::mRecordMode](#), and [AAX_TransportStateInfo_V1::mTransp](#)

15.282.2.2 operator"!=()

```
bool operator!= (
    const AAX_TransportStateInfo_V1 & state1,
    const AAX_TransportStateInfo_V1 & state2 ) [inline]
```

15.283 AAX_TransportTypes.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2020-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #ifndef AAX_TransportTypes_h_
00022 #define AAX_TransportTypes_h_
00023 #pragma once
00024
00025 // AAX Includes
00026 #include "AAX.h"
00027
00028 // Standard Library Includes
00029 #include <string>
00030 #include <sstream>
00031
00032 #include AAX_ALIGN_FILE_BEGIN
00033 #include AAX_ALIGN_FILE_HOST
00034 #include AAX_ALIGN_FILE_END
00035
00039 struct AAX_TransportStateInfo_V1
00040 {
00041     AAX_ETransportState mTransportState;
00042     AAX_ERecordMode mRecordMode;
00043     AAX_CBoolean mIsRecordEnabled;
00044     AAX_CBoolean mIsRecording;
00045     AAX_CBoolean mIsLoopEnabled;
00046
00047     AAX_TransportStateInfo_V1() :
00048         mTransportState(AAX_eTransportState_Unknown),
00049         mRecordMode(AAX_eRecordMode_Unknown),
00050         mIsRecordEnabled(false),
00051         mIsRecording(false),
00052         mIsLoopEnabled(false)
00053     {
00054         static_assert(sizeof(AAX_TransportStateInfo_V1) == 12, "Invalid size of
AAX_TransportStateInfo_V1 struct during compilation!");
00055     }
00056
00057     inline std::string ToString() const
00058     {
00059         std::stringstream ss;
00060
00061         ss << "{" << std::endl;
00062         ss << "\"transport_state\": " << mTransportState << ", " << std::endl;
00063         ss << "\"record_mode\": " << mRecordMode << ", " << std::endl;
00064         ss << "\"is_record_enabled\": " << mIsRecordEnabled << ", " << std::endl;
00065         ss << "\"is_recording\": " << mIsRecording << ", " << std::endl;
00066         ss << "\"is_loop_enabled\": " << mIsLoopEnabled << std::endl;
00067         ss << "}";
00068
00069         return ss.str();
00070     }
00071 };
00072
00073 #include AAX_ALIGN_FILE_BEGIN
00074 #include AAX_ALIGN_FILE_RESET
00075 #include AAX_ALIGN_FILE_END
00076
00077 inline bool operator==(const AAX_TransportStateInfo_V1& state1, const AAX_TransportStateInfo_V1&
state2)
```

```

00078 {
00079     return (state1.mTransportState == state2.mTransportState) && (state1.mRecordMode ==
state2.mRecordMode) &&
00080     (state1.mIsRecordEnabled == state2.mIsRecordEnabled) && (state1.mIsRecording ==
state2.mIsRecording) &&
00081     (state1.mIsLoopEnabled == state2.mIsLoopEnabled);
00082 }
00083
00084 inline bool operator!=(const AAX_TransportStateInfo_V1& state1, const AAX_TransportStateInfo_V1&
state2)
00085 {
00086     return !(state1 == state2);
00087 }
00088
00089 #endif // #ifndef AAX_TransportTypes_h_

```

15.284 AAX_UIDs.h File Reference

```

#include "acfbasetypes.h"
#include "defineacfuid.h"
#include "acfuuids.h"

```

15.284.1 Description

Unique identifiers for AAX/ACF interfaces.

Variables

AAX Host interface IDs

- const [acfiID AAXCompID_HostServices](#)
ACF component ID for AAX_IHostServices components.
- const [acfiID IID_IAAXHostServicesV1](#)
ACF interface ID for AAX_IACFHostServices.
- const [acfiID IID_IAAXHostServicesV2](#)
ACF interface ID for AAX_IACFHostServices_V2.
- const [acfiID IID_IAAXHostServicesV3](#)
ACF interface ID for AAX_IACFHostServices_V3.
- const [acfiID AAXCompID_AAXCollection](#)
ACF component ID for AAX_ICollection components.
- const [acfiID IID_IAAXCollectionV1](#)
ACF interface ID for AAX_IACFCollection.
- const [acfiID AAXCompID_AAXEffectDescriptor](#)
ACF component ID for AAX_IEffectDescriptor components.
- const [acfiID IID_IAAXEffectDescriptorV1](#)
ACF interface ID for AAX_IACFEffectDescriptor.
- const [acfiID IID_IAAXEffectDescriptorV2](#)
ACF interface ID for AAX_IACFEffectDescriptor_V2.
- const [acfiID AAXCompID_AAXComponentDescriptor](#)
ACF component ID for AAX_IComponentDescriptor components.
- const [acfiID IID_IAAXComponentDescriptorV1](#)
ACF interface ID for AAX_IACFComponentDescriptor.
- const [acfiID IID_IAAXComponentDescriptorV2](#)
ACF interface ID for AAX_IACFComponentDescriptor_V2.
- const [acfiID IID_IAAXComponentDescriptorV3](#)
ACF interface ID for AAX_IACFComponentDescriptor_V3.
- const [acfiID AAXCompID_AAXPropertyMap](#)

- ACF component ID for [AAX_IPropertyMap](#) components.
- const [acfiID IID_IAAXPropertyMapV1](#)
ACF interface ID for [AAX_IACFPropertyMap](#).
- const [acfiID IID_IAAXPropertyMapV2](#)
ACF interface ID for [AAX_IACFPropertyMap_V2](#).
- const [acfiID IID_IAAXPropertyMapV3](#)
ACF interface ID for [AAX_IACFPropertyMap_V3](#).
- const [acfiID AAXCompID_HostProcessorDelegate](#)
ACF component ID for [AAX_IHostProcessorDelegate](#) components.
- const [acfiID IID_IAAXHostProcessorDelegateV1](#)
ACF interface ID for [AAX_IACFHostProcessorDelegate](#).
- const [acfiID IID_IAAXHostProcessorDelegateV2](#)
ACF interface ID for [AAX_IACFHostProcessorDelegate_V2](#).
- const [acfiID IID_IAAXHostProcessorDelegateV3](#)
ACF interface ID for [AAX_IACFHostProcessorDelegate_V3](#).
- const [acfiID AAXCompID_AutomationDelegate](#)
ACF component ID for [AAX_IAutomationDelegate](#) components.
- const [acfiID IID_IAAXAutomationDelegateV1](#)
ACF interface ID for [AAX_IACFAutomationDelegate](#).
- const [acfiID AAXCompID_Controller](#)
ACF component ID for [AAX_IController](#) components.
- const [acfiID IID_IAAXControllerV1](#)
ACF interface ID for [AAX_IACFController](#).
- const [acfiID IID_IAAXControllerV2](#)
ACF interface ID for [AAX_IACFController_V2](#).
- const [acfiID IID_IAAXControllerV3](#)
ACF interface ID for [AAX_IACFController_V3](#).
- const [acfiID AAXCompID_PageTableController](#)
ACF component ID for [AAX](#) page table controller components.
- const [acfiID IID_IAAXPageTableController](#)
ACF interface ID for [AAX_IACFPageTableController](#).
- const [acfiID IID_IAAXPageTableControllerV2](#)
ACF interface ID for [AAX_IACFPageTableController_V2](#).
- const [acfiID AAXCompID_PrivateDataAccess](#)
ACF component ID for [AAX_IPrivateDataAccess](#) components.
- const [acfiID IID_IAAXPrivateDataAccessV1](#)
ACF interface ID for [AAX_IACFPrivateDataAccess](#).
- const [acfiID AAXCompID_ViewContainer](#)
ACF component ID for [AAX_IViewContainer](#) components.
- const [acfiID IID_IAAXViewContainerV1](#)
ACF interface ID for [AAX_IACFViewContainer](#).
- const [acfiID IID_IAAXViewContainerV2](#)
ACF interface ID for [AAX_IACFViewContainer_V2](#).
- const [acfiID IID_IAAXViewContainerV3](#)
ACF interface ID for [AAX_IACFViewContainer_V3](#).
- const [acfiID AAXCompID_Transport](#)
ACF component ID for [AAX_ITransport](#) components.
- const [acfiID IID_IAAXTransportV1](#)
ACF interface ID for [AAX_IACFTransport](#).
- const [acfiID IID_IAAXTransportV2](#)
ACF interface ID for [AAX_IACFTransport_V2](#).
- const [acfiID IID_IAAXTransportV3](#)
ACF interface ID for [AAX_IACFTransport_V3](#).
- const [acfiID IID_IAAXTransportV4](#)
ACF interface ID for [AAX_IACFTransport_V4](#).
- const [acfiID AAXCompID_TransportControl](#)
ACF component ID for [AAX_ITransportControl](#) components (accessed via [AAX_ITransport](#))
- const [acfiID IID_IAAXTransportControlV1](#)
ACF interface ID for [AAX_IACFTransportControl](#).

- const [acfiID AAXCompID_PageTable](#)
ACF component ID for [AAX_IPageTable](#) components.
- const [acfiID IID_IAAXPageTableV1](#)
ACF interface ID for [AAX_IACFPageTable](#).
- const [acfiID IID_IAAXPageTableV2](#)
ACF interface ID for [AAX_IACFPageTable_V2](#).
- const [acfiID AAX_CompID_DescriptionHost](#)
ACF component ID for [AAX_IDescriptionHost](#) components.
- const [acfiID IID_IAAXDescriptionHostV1](#)
ACF interface ID for [AAX_IACFDescriptionHost](#).
- const [acfiID AAX_CompID_FeatureInfo](#)
ACF component ID for [AAX_IFeatureInfo](#) components.
- const [acfiID IID_IAAXFeatureInfoV1](#)
ACF interface ID for [AAX_IACFFeatureInfo](#).
- const [acfiID AAXCompID_Task](#)
ACF component ID for [AAX_ITask](#) components.
- const [acfiID IID_IAAXTaskV1](#)
ACF interface ID for [AAX_IACFTask](#).
- const [acfiID AAXCompID_SessionDocument](#)
ACF component ID for [AAX_ISessionDocument](#) components.
- const [acfiID IID_IAAXSessionDocumentV1](#)
ACF interface ID for [AAX_IACFSessionDocument](#).

AAX plug-in interface IDs

- const [acfiID AAXCompID_EffectParameters](#)
ACF component ID for [AAX_IEffectParameters](#) components.
- const [acfiID IID_IAAXEffectParametersV1](#)
ACF interface ID for [AAX_IACFEffectParameters](#).
- const [acfiID IID_IAAXEffectParametersV2](#)
ACF interface ID for [AAX_IACFEffectParameters_V2](#).
- const [acfiID IID_IAAXEffectParametersV3](#)
ACF interface ID for [AAX_IACFEffectParameters_V3](#).
- const [acfiID IID_IAAXEffectParametersV4](#)
ACF interface ID for [AAX_IACFEffectParameters_V4](#).
- const [acfiID AAXCompID_HostProcessor](#)
ACF component ID for [AAX_IHostProcessor](#) components.
- const [acfiID IID_IAAXHostProcessorV1](#)
ACF interface ID for [AAX_IACFHostProcessor](#).
- const [acfiID IID_IAAXHostProcessorV2](#)
ACF interface ID for [AAX_IACFHostProcessor_V2](#).
- const [acfiID AAXCompID_EffectGUI](#)
ACF component ID for [AAX_IEffectGUI](#) components.
- const [acfiID IID_IAAXEffectGUIV1](#)
ACF interface ID for [AAX_IACFEffectGUI](#).
- const [acfiID AAXCompID_EffectDirectData](#)
ACF component ID for [AAX_IEffectDirectData](#) components.
- const [acfiID IID_IAAXEffectDirectDataV1](#)
ACF interface ID for [AAX_IACFEffectDirectData](#).
- const [acfiID IID_IAAXEffectDirectDataV2](#)
- const [acfiID AAXCompID_TaskAgent](#)
ACF component ID for [AAX_ITaskAgent](#) components.
- const [acfiID IID_IAAXTaskAgentV1](#)
ACF interface ID for [AAX_IACFTaskAgent](#).
- const [acfiID AAXCompID_SessionDocumentClient](#)
ACF component ID for [AAX_ISessionDocumentClient](#) components.
- const [acfiID IID_IAAXSessionDocumentClientV1](#)
ACF interface ID for [AAX_IACFSessionDocumentClient](#).

Other AAX interface IDs

- const [acfIID AAXCompID_DataBuffer](#)
ACF component ID for [AAX_IDataBuffer](#) components.
- const [acfIID IID_IAAXDataBufferV1](#)
ACF interface ID for [AAX_IACFDataBuffer](#).

AAX host attributes

- const [acfUID AAXATTR_Client_Level](#)
Client application level.

AAX Feature UIDs

- using [AAX_Feature_UID](#) = [acfUID](#)
- const [AAX_Feature_UID AAXATTR_ClientFeature_StemFormat](#)
Client stem format feature support.
- const [AAX_Feature_UID AAXATTR_ClientFeature_AuxOutputStem](#)
*Client *Auxiliary Output Stem* feature support.*
- const [AAX_Feature_UID AAXATTR_ClientFeature_SideChainInput](#)
- const [AAX_Feature_UID AAXATTR_ClientFeature_MIDI](#)
*Client *MIDI* feature support.*

AAX document data type UIDs

- using [AAX_DocumentData_UID](#) = [acfUID](#)
- const [AAX_DocumentData_UID AAX_DocumentDataType_TempoMap](#)

15.284.2 Typedef Documentation

15.284.2.1 AAX_Feature_UID

using [AAX_Feature_UID](#) = [acfUID](#)

Identifier for AAX features

See [AAX_IDescriptionHost::AcquireFeatureProperties\(\)](#) and [AAX_IFeatureInfo](#)

15.284.2.2 AAX_DocumentData_UID

using [AAX_DocumentData_UID](#) = [acfUID](#)

Identifier for AAX document data types

See also

[AAX_IACFSessionDocument](#)

15.284.3 Variable Documentation

15.284.3.1 AAXCompID_HostServices

```
const acfIID AAXCompID_HostServices
```

ACF component ID for [AAX_IHostServices](#) components.

15.284.3.2 IID_IAAXHostServicesV1

```
const acfIID IID_IAAXHostServicesV1
```

ACF interface ID for [AAX_IACFHostServices](#).

15.284.3.3 IID_IAAXHostServicesV2

```
const acfIID IID_IAAXHostServicesV2
```

ACF interface ID for [AAX_IACFHostServices_V2](#).

15.284.3.4 IID_IAAXHostServicesV3

```
const acfIID IID_IAAXHostServicesV3
```

ACF interface ID for [AAX_IACFHostServices_V3](#).

15.284.3.5 AAXCompID_AAXCollection

```
const acfIID AAXCompID_AAXCollection
```

ACF component ID for [AAX_ICollection](#) components.

15.284.3.6 IID_IAAXCollectionV1

```
const acfIID IID_IAAXCollectionV1
```

ACF interface ID for [AAX_IACFCollection](#).

15.284.3.7 AAXCompID_AAXEffectDescriptor

```
const acfIID AAXCompID_AAXEffectDescriptor
```

ACF component ID for [AAX_IEffectDescriptor](#) components.

15.284.3.8 IID_IAAXEffectDescriptorV1

```
const acfIID IID_IAAXEffectDescriptorV1
```

ACF interface ID for [AAX_IACFEffectDescriptor](#).

15.284.3.9 IID_IAAXEffectDescriptorV2

```
const acfIID IID_IAAXEffectDescriptorV2
```

ACF interface ID for [AAX_IACFEffectDescriptor_V2](#).

15.284.3.10 AAXCompID_AAXComponentDescriptor

```
const acfIID AAXCompID_AAXComponentDescriptor
```

ACF component ID for [AAX_IComponentDescriptor](#) components.

15.284.3.11 IID_IAAXComponentDescriptorV1

```
const acfIID IID_IAAXComponentDescriptorV1
```

ACF interface ID for [AAX_IACFComponentDescriptor](#).

15.284.3.12 IID_IAAXComponentDescriptorV2

```
const acfIID IID_IAAXComponentDescriptorV2
```

ACF interface ID for [AAX_IACFComponentDescriptor_V2](#).

15.284.3.13 IID_IAAXComponentDescriptorV3

```
const acfIID IID_IAAXComponentDescriptorV3
```

ACF interface ID for [AAX_IACFComponentDescriptor_V3](#).

15.284.3.14 AAXCompID_AAXPropertyMap

```
const acfIID AAXCompID_AAXPropertyMap
```

ACF component ID for [AAX_IPropertyMap](#) components.

15.284.3.15 IID_IAAXPropertyMapV1

```
const acfIID IID_IAAXPropertyMapV1
```

ACF interface ID for [AAX_IACFPropertyMap](#).

15.284.3.16 IID_IAAXPropertyMapV2

```
const acfIID IID_IAAXPropertyMapV2
```

ACF interface ID for [AAX_IACFPropertyMap_V2](#).

15.284.3.17 IID_IAAXPropertyMapV3

```
const acfIID IID_IAAXPropertyMapV3
```

ACF interface ID for [AAX_IACFPropertyMap_V3](#).

15.284.3.18 AAXCompID_HostProcessorDelegate

```
const acfIID AAXCompID_HostProcessorDelegate
```

ACF component ID for [AAX_IHostProcessorDelegate](#) components.

15.284.3.19 IID_IAAXHostProcessorDelegateV1

```
const acfIID IID_IAAXHostProcessorDelegateV1
```

ACF interface ID for [AAX_IACFHostProcessorDelegate](#).

15.284.3.20 IID_IAAXHostProcessorDelegateV2

```
const acfIID IID_IAAXHostProcessorDelegateV2
```

ACF interface ID for [AAX_IACFHostProcessorDelegate_V2](#).

15.284.3.21 IID_IAAXHostProcessorDelegateV3

```
const acfIID IID_IAAXHostProcessorDelegateV3
```

ACF interface ID for [AAX_IACFHostProcessorDelegate_V3](#).

15.284.3.22 AAXCompID_AutomationDelegate

```
const acfIID AAXCompID_AutomationDelegate
```

ACF component ID for [AAX_IAutomationDelegate](#) components.

15.284.3.23 IID_IAAXAutomationDelegateV1

```
const acfIID IID_IAAXAutomationDelegateV1
```

ACF interface ID for [AAX_IACFAutomationDelegate](#).

15.284.3.24 AAXCompID_Controller

```
const acfIID AAXCompID_Controller
```

ACF component ID for [AAX_IController](#) components.

15.284.3.25 IID_IAAXControllerV1

```
const acfIID IID_IAAXControllerV1
```

ACF interface ID for [AAX_IACFController](#).

15.284.3.26 IID_IAAXControllerV2

```
const acfIID IID_IAAXControllerV2
```

ACF interface ID for [AAX_IACFController_V2](#).

15.284.3.27 IID_IAAXControllerV3

```
const acfIID IID_IAAXControllerV3
```

ACF interface ID for [AAX_IACFController_V3](#).

15.284.3.28 AAXCompID_PageTableController

```
const acfIID AAXCompID_PageTableController
```

ACF component ID for AAX page table controller components.

15.284.3.29 IID_IAAXPageTableController

```
const acfIID IID_IAAXPageTableController
```

ACF interface ID for [AAX_IACFPageTableController](#).

15.284.3.30 IID_IAAXPageTableControllerV2

```
const acfIID IID_IAAXPageTableControllerV2
```

ACF interface ID for [AAX_IACFPPageTableController_V2](#).

15.284.3.31 AAXCompID_PrivateDataAccess

```
const acfIID AAXCompID_PrivateDataAccess
```

ACF component ID for [AAX_IPrivateDataAccess](#) components.

15.284.3.32 IID_IAAXPrivateDataAccessV1

```
const acfIID IID_IAAXPrivateDataAccessV1
```

ACF interface ID for [AAX_IACFPrivateDataAccess](#).

15.284.3.33 AAXCompID_ViewContainer

```
const acfIID AAXCompID_ViewContainer
```

ACF component ID for [AAX_IViewContainer](#) components.

15.284.3.34 IID_IAAXViewContainerV1

```
const acfIID IID_IAAXViewContainerV1
```

ACF interface ID for [AAX_IACFViewContainer](#).

15.284.3.35 IID_IAAXViewContainerV2

```
const acfIID IID_IAAXViewContainerV2
```

ACF interface ID for [AAX_IACFViewContainer_V2](#).

15.284.3.36 IID_IAAXViewContainerV3

```
const acfIID IID_IAAXViewContainerV3
```

ACF interface ID for [AAX_IACFViewContainer_V3](#).

15.284.3.37 AAXCompID_Transport

```
const acfIID AAXCompID_Transport
```

ACF component ID for [AAX_ITransport](#) components.

15.284.3.38 IID_IAAXTransportV1

```
const acfIID IID_IAAXTransportV1
```

ACF interface ID for [AAX_IACFTransport](#).

15.284.3.39 IID_IAAXTransportV2

```
const acfIID IID_IAAXTransportV2
```

ACF interface ID for [AAX_IACFTransport_V2](#).

15.284.3.40 IID_IAAXTransportV3

```
const acfIID IID_IAAXTransportV3
```

ACF interface ID for [AAX_IACFTransport_V3](#).

15.284.3.41 IID_IAAXTransportV4

```
const acfIID IID_IAAXTransportV4
```

ACF interface ID for [AAX_IACFTransport_V4](#).

15.284.3.42 AAXCompID_TransportControl

```
const acfIID AAXCompID_TransportControl
```

ACF component ID for AAX_ITransportControl components (accessed via [AAX_ITransport](#))

15.284.3.43 IID_IAAXTransportControlV1

```
const acfIID IID_IAAXTransportControlV1
```

ACF interface ID for [AAX_IACFTransportControl](#).

15.284.3.44 AAXCompID_PageTable

```
const acfIID AAXCompID_PageTable
```

ACF component ID for [AAX_IPageTable](#) components.

15.284.3.45 IID_IAAXPageTableV1

```
const acfIID IID_IAAXPageTableV1
```

ACF interface ID for [AAX_IACFPageTable](#).

15.284.3.46 IID_IAAXPageTableV2

```
const acfIID IID_IAAXPageTableV2
```

ACF interface ID for [AAX_IACFPageTable_V2](#).

15.284.3.47 AAX_CompID_DescriptionHost

```
const acfIID AAX_CompID_DescriptionHost
```

ACF component ID for [AAX_IDescriptionHost](#) components.

15.284.3.48 IID_IAAXDescriptionHostV1

```
const acfIID IID_IAAXDescriptionHostV1
```

ACF interface ID for [AAX_IACFDescriptionHost](#).

15.284.3.49 AAX_CompID_FeatureInfo

```
const acfIID AAX_CompID_FeatureInfo
```

ACF component ID for [AAX_IFeatureInfo](#) components.

15.284.3.50 IID_IAAXFeatureInfoV1

```
const acfIID IID_IAAXFeatureInfoV1
```

ACF interface ID for [AAX_IACFFeatureInfo](#).

15.284.3.51 AAXCompID_Task

```
const acfIID AAXCompID_Task
```

ACF component ID for [AAX_ITask](#) components.

15.284.3.52 IID_IAAXTaskV1

```
const acfIID IID_IAAXTaskV1
```

ACF interface ID for [AAX_IACFTask](#).

15.284.3.53 AAXCompID_SessionDocument

```
const acfIID AAXCompID_SessionDocument
```

ACF component ID for [AAX_ISessionDocument](#) components.

15.284.3.54 IID_IAAXSessionDocumentV1

```
const acfIID IID_IAAXSessionDocumentV1
```

ACF interface ID for [AAX_IACFSessionDocument](#).

15.284.3.55 AAXCompID_EffectParameters

```
const acfIID AAXCompID_EffectParameters
```

ACF component ID for [AAX_IEffectParameters](#) components.

15.284.3.56 IID_IAAXEffectParametersV1

```
const acfIID IID_IAAXEffectParametersV1
```

ACF interface ID for [AAX_IACFEffectParameters](#).

15.284.3.57 IID_IAAXEffectParametersV2

```
const acfIID IID_IAAXEffectParametersV2
```

ACF interface ID for [AAX_IACFEffectParameters_V2](#).

15.284.3.58 IID_IAAXEffectParametersV3

```
const acfIID IID_IAAXEffectParametersV3
```

ACF interface ID for [AAX_IACFEffectParameters_V3](#).

15.284.3.59 IID_IAAXEffectParametersV4

```
const acfIID IID_IAAXEffectParametersV4
```

ACF interface ID for [AAX_IACFEffectParameters_V4](#).

15.284.3.60 AAXCompID_HostProcessor

```
const acfIID AAXCompID_HostProcessor
```

ACF component ID for [AAX_IHostProcessor](#) components.

15.284.3.61 IID_IAAXHostProcessorV1

```
const acfIID IID_IAAXHostProcessorV1
```

ACF interface ID for [AAX_IACFHostProcessor](#).

15.284.3.62 IID_IAAXHostProcessorV2

```
const acfIID IID_IAAXHostProcessorV2
```

ACF interface ID for [AAX_IACFHostProcessor_V2](#).

15.284.3.63 AAXCompID_EffectGUI

```
const acfIID AAXCompID_EffectGUI
```

ACF component ID for [AAX_IEffectGUI](#) components.

15.284.3.64 IID_IAAXEffectGUIV1

```
const acfIID IID_IAAXEffectGUIV1
```

ACF interface ID for [AAX_IACFEffectGUI](#).

15.284.3.65 AAXCompID_EffectDirectData

```
const acfIID AAXCompID_EffectDirectData
```

ACF component ID for [AAX_IEffectDirectData](#) components.

15.284.3.66 IID_IAAXEffectDirectDataV1

```
const acfIID IID_IAAXEffectDirectDataV1
```

ACF interface ID for [AAX_IACFEffectDirectData](#).

15.284.3.67 IID_IAAXEffectDirectDataV2

```
const acfIID IID_IAAXEffectDirectDataV2
```

15.284.3.68 AAXCompID_TaskAgent

```
const acfIID AAXCompID_TaskAgent
```

ACF component ID for [AAX_ITaskAgent](#) components.

15.284.3.69 IID_IAAXTaskAgentV1

```
const acfIID IID_IAAXTaskAgentV1
```

ACF interface ID for [AAX_IACFTaskAgent](#).

15.284.3.70 AAXCompID_SessionDocumentClient

```
const acfIID AAXCompID_SessionDocumentClient
```

ACF component ID for [AAX_ISessionDocumentClient](#) components.

15.284.3.71 IID_IAAXSessionDocumentClientV1

```
const acfIID IID_IAAXSessionDocumentClientV1
```

ACF interface ID for [AAX_IACFSessionDocumentClient](#).

15.284.3.72 AAXCompID_DataBuffer

```
const acfIID AAXCompID_DataBuffer
```

ACF component ID for [AAX_IDataBuffer](#) components.

15.284.3.73 IID_IAAXDataBufferV1

```
const acfIID IID_IAAXDataBufferV1
```

ACF interface ID for [AAX_IACFDataBuffer](#).

15.284.3.74 AAXATTR_ClientFeature_StemFormat

```
AAXATTR_ClientFeature_StemFormat
```

Client stem format feature support.

To determine the client's support for specific stem formats, use the property map

Property map contents Key: [AAX_EStemFormat](#) values Value: [AAX_ESupportLevel](#) value; if undefined then no information is available

15.284.3.75 AAXATTR_ClientFeature_AuxOutputStem

```
AAXATTR_ClientFeature_AuxOutputStem
```

Client [Auxiliary Output Stem](#) feature support.

Client [Side Chain](#) feature support.

Plug-ins must detect when a host does not support AOS in order to avoid running off the end of the output audio buffer list in the audio algorithm.

[AddAuxOutputStem\(\)](#) will return an error for hosts that do not support this feature, so typically a feature support query using this [AAX_Feature_UID](#) is not required.

15.284.3.76 AAXATTR_ClientFeature_SideChainInput

```
const AAX_Feature_UID AAXATTR_ClientFeature_SideChainInput
```

15.284.3.77 AAXATTR_ClientFeature_MIDI

AAXATTR_ClientFeature_MIDI

Client [MIDI](#) feature support.

15.284.3.78 AAXATTR_Client_Level

AAXATTR_Client_Level

Client application level.

Type: uint32_t (ACFTypeID_UInt32) Value: one of [AAX_EHostLevel](#)

Query using the host's [IACFDefinition](#)

15.284.3.79 AAX_DocumentDataType_TempoMap

AAX_DocumentDataType_TempoMap

The session tempo map

Provides an [AAX_IACFDataBuffer](#) containing a list of [AAX_CTempoBreakpoint](#) elements.

15.285 AAX_UIDs.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00022 #ifndef AAX_UIDS_H
00023 #define AAX_UIDS_H
00025
00026 #include "acfbasetypes.h"
00027 #include "defineacfluid.h"
00028
00029 // Pull in the declarations of all standard ACF UUIDs
00030 #include "acfluids.h"
00031
00032
00033
00039 DEFINE_ACFUID(acfIID, AAXCompID_HostServices, 0x88882c2d, 0xebbc, 0x42ef, 0xc0, 0xab, 0x89, 0x81,
0xb0, 0xbd, 0x0c, 0xca);
00041 DEFINE_ACFUID(acfIID, IID_IAAXHostServicesV1, 0x96d42c2d, 0xebbc, 0x41ef, 0xb0, 0xab, 0x99, 0x91,
0xa0, 0xed, 0x0c, 0xca);
00043 DEFINE_ACFUID(acfIID, IID_IAAXHostServicesV2, 0xa207ee9e, 0xb442, 0x11e4, 0xa7, 0x1e, 0x12, 0xe3,
0xf5, 0x12, 0xa3, 0x38);
00045 DEFINE_ACFUID(acfIID, IID_IAAXHostServicesV3, 0x12bea399, 0x9a4f, 0x4353, 0x80, 0x98, 0x39, 0x16,
0xfa, 0x71, 0x89, 0x8d);
00046
```

```
00048 DEFINE_ACFUID(acfIID, AAXCompID_AAXCollection, 0x89882c2d, 0x77bc, 0x42ef, 0x70, 0x7b, 0x79, 0x81,
0xb7, 0xbd, 0x0c, 0xca);
00050 DEFINE_ACFUID(acfIID, IID_IAAXCollectionV1, 0x96d42c2d, 0xebbc, 0x41df, 0xb1, 0xab, 0x99, 0x91, 0xa2,
0xee, 0x0c, 0xca);
00051
00053 DEFINE_ACFUID(acfIID, AAXCompID_AAXEffectDescriptor, 0x89872c2d, 0x75bc, 0x423f, 0x40, 0x1b, 0xf9,
0xa1, 0xba, 0xad, 0x0c, 0xca);
00055 DEFINE_ACFUID(acfIID, IID_IAAXEffectDescriptorV1, 0x96d42c2d, 0xebbc, 0x41ef, 0xd1, 0xcb, 0x49, 0x94,
0x42, 0xe4, 0x0f, 0xda);
00057 DEFINE_ACFUID(acfIID, IID_IAAXEffectDescriptorV2, 0x41eccc52, 0x416b, 0x4072, 0x84, 0xbd, 0x40, 0xb0,
0x52, 0x10, 0xa7, 0x4c);
00058
00060 DEFINE_ACFUID(acfIID, AAXCompID_AAXComponentDescriptor, 0x94872c3d, 0x95bc, 0x413d, 0xd0, 0xdb, 0xd9,
0xb1, 0x2a, 0xad, 0x0c, 0xca);
00062 DEFINE_ACFUID(acfIID, IID_IAAXComponentDescriptorV1, 0x96e42c2d, 0xe2bc, 0x51ef, 0x61, 0xc7, 0x48,
0x99, 0x4a, 0xeb, 0x0c, 0xda);
00064 DEFINE_ACFUID(acfIID, IID_IAAXComponentDescriptorV2, 0x1895259e, 0xaa9, 0x4f0f, 0xa9, 0x85, 0x14,
0x98, 0x37, 0xb7, 0x6f, 0x89);
00066 DEFINE_ACFUID(acfIID, IID_IAAXComponentDescriptorV3, 0x979cfc4d, 0x2bcb, 0x43ae, 0x9c, 0xd0, 0x2d,
0x0e, 0xcd, 0x2a, 0xdc, 0xd5);
00067
00068
00070 DEFINE_ACFUID(acfIID, AAXCompID_AAXPropertyMap, 0xa587ad3d, 0xd53c, 0x4adc, 0xd0, 0xdd, 0xd9, 0xd1,
0x2d, 0xdd, 0xdc, 0xda);
00072 DEFINE_ACFUID(acfIID, IID_IAAXPropertyMapV1, 0x96ee2c2d, 0xeec, 0x5eff, 0xe2, 0xd7, 0xe8, 0x49, 0xe3,
0xe2, 0xee, 0xee);
00074 DEFINE_ACFUID(acfIID, IID_IAAXPropertyMapV2, 0x7177df80, 0x7c9c, 0x11e2, 0xb9, 0x2a, 0x08, 0x00, 0x20,
0x0c, 0x9a, 0x66);
00076 DEFINE_ACFUID(acfIID, IID_IAAXPropertyMapV3, 0x6d4ab208, 0xd34b, 0x4368, 0xb4, 0xf1, 0x58, 0xbc, 0x24,
0x3f, 0x45, 0xc9);
00077
00079 DEFINE_ACFUID(acfIID, AAXCompID_HostProcessorDelegate, 0xab933d9d, 0x5434, 0x25dc, 0x19, 0x0b, 0x09,
0x23, 0x2d, 0xdd, 0x38, 0x8a);
00081 DEFINE_ACFUID(acfIID, IID_IAAXHostProcessorDelegateV1, 0x9d4e3d3d, 0x43dc, 0x5eda, 0x82, 0x27, 0xe2,
0xf2, 0xf8, 0xd5, 0x6e, 0x8e);
00083 DEFINE_ACFUID(acfIID, IID_IAAXHostProcessorDelegateV2, 0xfb6de2c9, 0x29d0, 0x4683, 0xb3, 0x48, 0xc7,
0x78, 0xf4, 0xcd, 0x62, 0x5b);
00085 DEFINE_ACFUID(acfIID, IID_IAAXHostProcessorDelegateV3, 0x5dfef2b3, 0x7027, 0x46b5, 0xae, 0x3a, 0x27,
0x94, 0xc6, 0xe0, 0xa8, 0xa0);
00086
00088 DEFINE_ACFUID(acfIID, AAXCompID_AutomationDelegate, 0xab943d9d, 0x5534, 0x26dc, 0x29, 0xab, 0xc9,
0xb3, 0x2d, 0x2d, 0x28, 0x8a);
00090 DEFINE_ACFUID(acfIID, IID_IAAXAutomationDelegateV1, 0x9d5e3d3d, 0x42dc, 0x5efa, 0x22, 0x17, 0xee,
0xe2, 0xe8, 0xe5, 0x3e, 0x2e);
00091
00093 DEFINE_ACFUID(acfIID, AAXCompID_Controller, 0xab944d4d, 0x15c4, 0xc61c, 0x3d, 0x3b, 0xf9, 0xbf, 0x1d,
0x20, 0x18, 0x4a);
00095 DEFINE_ACFUID(acfIID, IID_IAAXControllerV1, 0x9d5e3e3d, 0x52dc, 0x5efb, 0x20, 0x18, 0xde, 0x1d, 0xe2,
0xe6, 0x3f, 0x4e);
00097 DEFINE_ACFUID(acfIID, IID_IAAXControllerV2, 0x4c59aa0e, 0xd7c0, 0x4205, 0x8b, 0x6c, 0x32, 0x46, 0x8d,
0x42, 0xd2, 0x02);
00099 DEFINE_ACFUID(acfIID, IID_IAAXControllerV3, 0xdd6f168c, 0xda86, 0x44f8, 0xb8, 0x64, 0xd6, 0xcd, 0x22,
0x19, 0x26, 0xe7);
00100
00102 DEFINE_ACFUID(acfIID, AAXCompID_PageTableController, 0x63355d80, 0xbfe1, 0x4291, 0xa6, 0x27, 0xc6,
0x5c, 0xb9, 0x58, 0x91, 0x40);
00104 DEFINE_ACFUID(acfIID, IID_IAAXPageTableController, 0x2e9d35fb, 0xbacc, 0x4b2c, 0xb5, 0xd7, 0xc6, 0xe8,
0x51, 0xf5, 0x69, 0xbd);
00106 DEFINE_ACFUID(acfIID, IID_IAAXPageTableControllerV2, 0x6c6b83e, 0x9d87, 0x4938, 0x8a, 0x38, 0xf8,
0xe4, 0x5b, 0x10, 0xa2, 0x4a);
00107
00109 DEFINE_ACFUID(acfIID, AAXCompID_PrivateDataAccess, 0xab945d4d, 0x15c6, 0xc61c, 0x3f, 0x3f, 0xf9, 0xbf,
0x1d, 0x20, 0x18, 0x4c);
00111 DEFINE_ACFUID(acfIID, IID_IAAXPrivateDataAccessV1, 0x9d5e6e3f, 0x52de, 0x5efd, 0x22, 0x18, 0xdf, 0x1f,
0xe3, 0xe8, 0x3f, 0x4c);
00112
00114 DEFINE_ACFUID(acfIID, AAXCompID_ViewContainer, 0xdede24bd, 0xc2ff, 0x467a, 0xae, 0x2d, 0x5f, 0x29,
0x1d, 0x19, 0x22, 0x2b);
00116 DEFINE_ACFUID(acfIID, IID_IAAXViewContainerV1, 0x22da0bbc, 0xd550, 0x4d5e, 0x8c, 0xc6, 0x73, 0x44,
0x83, 0xb8, 0x83, 0x7f);
00118 DEFINE_ACFUID(acfIID, IID_IAAXViewContainerV2, 0x9143a0be, 0x7a79, 0x4d02, 0xae, 0x25, 0xaa, 0xdb,
0xa7, 0x6a, 0x50, 0xb2);
00120 DEFINE_ACFUID(acfIID, IID_IAAXViewContainerV3, 0x07cda0fd, 0xbe98, 0x4dd7, 0x92, 0xe0, 0x02, 0x37,
0x57, 0xdf, 0x2e, 0x01);
00121
00123 DEFINE_ACFUID(acfIID, AAXCompID_Transport, 0xa9fa236, 0x2176, 0x49e1, 0xb6, 0x24, 0x82, 0x7d, 0x2b,
0x43, 0x31, 0x5c);
00125 DEFINE_ACFUID(acfIID, IID_IAAXTransportV1, 0x5cee4ef4, 0x6337, 0x4359, 0xb6, 0x3b, 0xfe, 0x58, 0xdc,
0x36, 0x54, 0x3a);
00127 DEFINE_ACFUID(acfIID, IID_IAAXTransportV2, 0x203cbd9f, 0x982c, 0x4fe6, 0xa8, 0x27, 0x7, 0x48, 0x2,
0x57, 0xae, 0xc3);
00129 DEFINE_ACFUID(acfIID, IID_IAAXTransportV3, 0xaf79e815, 0xecfe, 0x1fb4, 0x8a, 0x2e, 0x24, 0xab, 0x0e,
0xc0, 0x8e, 0xf0);
00131 DEFINE_ACFUID(acfIID, IID_IAAXTransportV4, 0xcad3748b, 0x5f34, 0x4a1d, 0xb5, 0x9e, 0x12, 0x6e, 0xcf,
0xb0, 0x11, 0x77);
00132
00134 DEFINE_ACFUID(acfIID, AAXCompID_TransportControl, 0x0717ac4d, 0xdf87, 0x44b1, 0x82, 0x7b, 0x5b, 0x6f,
```

```
0x9b, 0xe3, 0x50, 0xf5);
00136 DEFINE_ACFUID(acfIID, IID_IAAXTransportControlV1, 0xce6ddb20, 0x1b7c, 0x4559, 0x9e, 0xe8, 0xd7, 0x69,
0x86, 0x45, 0xa1, 0x43);
00137
00139 DEFINE_ACFUID(acfIID, AAXCompID_PageTable, 0xdbc22879, 0xa24e, 0x4ac6, 0x97, 0x21, 0x93, 0x8b, 0x72,
0xd8, 0xe8, 0x1b);
00141 DEFINE_ACFUID(acfIID, IID_IAAXPageTableV1, 0x33c9e5be, 0x1ce3, 0x4085, 0x91, 0xa7, 0x09, 0xd6, 0xf8,
0xee, 0x4b, 0x64);
00143 DEFINE_ACFUID(acfIID, IID_IAAXPageTableV2, 0xd0f25d1b, 0x9c5b, 0x4d2e, 0x8f, 0x1f, 0x45, 0xbc, 0x93,
0x47, 0x32, 0xf7);
00144
00145
00147 DEFINE_ACFUID(acfIID, AAXCompID_DescriptionHost, 0x84e184ce, 0x353c, 0x4928, 0x80, 0x61, 0x04, 0x60,
0x06, 0xb3, 0x1b, 0x52);
00149 DEFINE_ACFUID(acfIID, IID_IAAXDescriptionHostV1, 0xe5bc71df, 0x4c1f, 0x4cc4, 0x81, 0x4a, 0x5a, 0x7d,
0xd0, 0xe7, 0x0e, 0xf5);
00150
00152 DEFINE_ACFUID(acfIID, AAXCompID_FeatureInfo, 0x617d2e4f, 0x3556, 0x483b, 0xb4, 0xde, 0x05, 0x3c,
0xc3, 0x92, 0x17, 0x53);
00154 DEFINE_ACFUID(acfIID, IID_IAAXFeatureInfoV1, 0x24545609, 0xa7c4, 0x44d4, 0xab, 0xb8, 0xcf, 0x13, 0xea,
0x9d, 0x0b, 0xdf);
00155
00157 DEFINE_ACFUID(acfIID, AAXCompID_Task, 0xa5237386, 0xd1a7, 0x490d, 0x5, 0x8, 0x3, 0x2, 0xd, 0x0, 0x2,
0x1);
00159 DEFINE_ACFUID(acfIID, IID_IAAXTaskV1, 0x9733f64b, 0x45d6, 0x47ba, 0x8, 0xb, 0x9, 0xd, 0xd, 0x7, 0x8,
0xa);
00160
00162 DEFINE_ACFUID(acfIID, AAXCompID_SessionDocument, 0x65fd4d4a, 0xf85e, 0x46fd, 0x8b, 0x7c, 0xa0, 0x31,
0x5c, 0x93, 0x2a, 0xd1);
00164 DEFINE_ACFUID(acfIID, IID_IAAXSessionDocumentV1, 0x4be26025, 0x27c9, 0x467e, 0x85, 0xd6, 0x78, 0xb5,
0xf1, 0xea, 0x7c, 0xdb);
00165
00167
00168
00169
00170
00171
00177 DEFINE_ACFUID(acfIID, AAXCompID_EffectParameters, 0xab97bd9d, 0x9b3c, 0x4bdc, 0xb9, 0x9b, 0x59, 0x51,
0xbd, 0x5d, 0x48, 0x4a);
00179 DEFINE_ACFUID(acfIID, IID_IAAXEffectParametersV1, 0x964e333d, 0x334c, 0x533f, 0xc2, 0xc7, 0x38, 0x34,
0xc3, 0xc2, 0x3e, 0x3e);
00181 DEFINE_ACFUID(acfIID, IID_IAAXEffectParametersV2, 0xf1f47d06, 0x308f, 0x4cc5, 0x9c, 0x7c, 0x50, 0xa8,
0x3f, 0x8a, 0xb8, 0x13);
00183 DEFINE_ACFUID(acfIID, IID_IAAXEffectParametersV3, 0xd2540e9d, 0x9163, 0x42bb, 0xa6, 0xfd, 0x81, 0xe1,
0xe, 0xa3, 0x24, 0x98);
00185 DEFINE_ACFUID(acfIID, IID_IAAXEffectParametersV4, 0x2e485536, 0x31a3, 0x4697, 0x9c, 0x16, 0xe5, 0x9b,
0xf6, 0xb2, 0x8a, 0x41);
00186
00188 DEFINE_ACFUID(acfIID, AAXCompID_HostProcessor, 0xab953d9d, 0x5b34, 0x45dc, 0x49, 0x3b, 0x29, 0x53,
0xcd, 0xdd, 0x48, 0x4a);
00190 DEFINE_ACFUID(acfIID, IID_IAAXHostProcessorV1, 0x964e3f3d, 0x434c, 0x5e3a, 0xa2, 0xe7, 0xe8, 0xf4,
0xf3, 0xd2, 0x2e, 0x2e);
00192 DEFINE_ACFUID(acfIID, IID_IAAXHostProcessorV2, 0x457546c0, 0xf6bc, 0x4af9, 0xbf, 0xf7, 0xeb, 0xdd,
0xc0, 0x5e, 0x56, 0xde);
00193
00195 DEFINE_ACFUID(acfIID, AAXCompID_EffectGUI, 0xab94339d, 0x3b34, 0x35dc, 0x29, 0x32, 0x19, 0x23, 0x1d,
0x1d, 0x48, 0x2a);
00197 DEFINE_ACFUID(acfIID, IID_IAAXEffectGUIV1, 0x964e323d, 0x424c, 0x5e1a, 0x22, 0x27, 0x28, 0x24, 0x23,
0x22, 0x2e, 0x1e);
00198
00200 DEFINE_ACFUID(acfIID, AAXCompID_EffectDirectData, 0xaafe80ab, 0x5b34, 0x4522, 0x49, 0x3b, 0x29, 0x53,
0xcd, 0xdd, 0x48, 0x4b);
00202 DEFINE_ACFUID(acfIID, IID_IAAXEffectDirectDataV1, 0x964e80ab, 0x434c, 0x5e22, 0xa2, 0xe7, 0xe8, 0xf4,
0xf3, 0xd2, 0x2e, 0x2f);
00203 // ACF interface ID for \ref AAX_IACFEEffectDirectData_V2
00204 DEFINE_ACFUID(acfIID, IID_IAAXEffectDirectDataV2, 0x156ea622, 0xbd2e, 0x11e9, 0x9c, 0xb5, 0x2a, 0x2a,
0xe2, 0xdb, 0xcc, 0xe4);
00205
00207 DEFINE_ACFUID(acfIID, AAXCompID_TaskAgent, 0xb0753064, 0xc37e, 0x11ed, 0xaf, 0xa1, 0x02, 0x42, 0xac,
0xe2, 0x00, 0x12);
00209 DEFINE_ACFUID(acfIID, IID_IAAXTaskAgentV1, 0xc096be3e, 0xbc3e, 0x4c38, 0x86, 0x1d, 0x06, 0xa4, 0xba,
0xa4, 0x10, 0x05);
00210
00212 DEFINE_ACFUID(acfIID, AAXCompID_SessionDocumentClient, 0x2280c3d5, 0x38f9, 0x43c5, 0x90, 0x1d, 0x8d,
0x1a, 0xfe, 0xb4, 0x2f, 0xa5);
00214 DEFINE_ACFUID(acfIID, IID_IAAXSessionDocumentClientV1, 0xadaebe77, 0xe1b6, 0x468d, 0x96, 0x60, 0xb6,
0xfb, 0xb7, 0x22, 0x4c, 0xa8);
00215
00216
00218
00219
00220
00226 DEFINE_ACFUID(acfIID, AAXCompID_DataBuffer, 0x2b21890c, 0x02c9, 0x4a56, 0xf, 0xc, 0xe, 0x3, 0x9, 0x3,
0x1, 0x6);
00228 DEFINE_ACFUID(acfIID, IID_IAAXDataBufferV1, 0x206ec31a, 0x7756, 0x4220, 0x6, 0x0, 0xc, 0xf, 0x6, 0xf,
0x7, 0x7);
00230
```



```

00231
00232
00233
00234
00235
00240
00245 using AAX_Feature_UID = acfUID;
00246
00258 DEFINE_ACFUID(AAX_Feature_UID, AAXATTR_ClientFeature_StemFormat, 0x729dd3e6, 0xd3dc, 0x484c, 0x91,
0x69, 0xf0, 0x64, 0xa0, 0x12, 0x60, 0x1d);
00259
00270 DEFINE_ACFUID(AAX_Feature_UID, AAXATTR_ClientFeature_AuxOutputStem, 0x5bea3f7a, 0x2be8, 0x4fe1, 0x83,
0xb2, 0x94, 0xec, 0x91, 0x31, 0xb8, 0x52);
00271
00276 DEFINE_ACFUID(AAX_Feature_UID, AAXATTR_ClientFeature_SideChainInput, 0x98b0a514, 0x2b96, 0x4e1f, 0x87,
0x81, 0x99, 0x08, 0xc9, 0xe3, 0xe6, 0x8b);
00277
00282 DEFINE_ACFUID(AAX_Feature_UID, AAXATTR_ClientFeature_MIDI, 0xf5b0816c, 0x5768, 0x49c2, 0xae, 0x3e,
0x85, 0x0d, 0xe3, 0x42, 0xeb, 0x07);
00283
00284
00286
00287
00292
00303 DEFINE_ACFUID(acfUID, AAXATTR_Client_Level, 0xe550868e, 0x1e6a, 0x482b, 0xb5, 0x86, 0x73, 0xf1, 0x24,
0x6e, 0x12, 0x6b);
00304
00306
00310
00315 using AAX_DocumentData_UID = acfUID;
00316
00324 DEFINE_ACFUID(AAX_DocumentData_UID, AAX_DocumentDataType_TempoMap, 0x2515e52b, 0x5b3e, 0x4354, 0x86,
0xeb, 0x93, 0x49, 0x6a, 0xc8, 0xa3, 0x37);
00325
00327
00329 #endif

```

15.286 AAX_UtilsNative.h File Reference

```

#include "AAX_CString.h"
#include "AAX_IString.h"
#include "AAX_Assert.h"
#include "AAX.h"
#include <cmath>
#include <string.h>

```

15.286.1 Description

Various utility definitions for AAX Native.

Namespaces

- namespace [AAX](#)

Macros

- #define [_AAX_UTILSNATIVE_H_](#)

Functions

- double [AAX::SafeLog](#) (double aValue)
Double-precision safe log function. Returns zero for input values that are ≤ 0.0 .
- float [AAX::SafeLogf](#) (float aValue)
Single-precision safe log function. Returns zero for input values that are ≤ 0.0 .
- [AAX_CBoolean AAX::IsParameterIDEqual](#) ([AAX_CParamID](#) iParam1, [AAX_CParamID](#) iParam2)
Helper function to check if two parameter IDs are equivalent.
- [AAX_CBoolean AAX::IsEffectIDEqual](#) (const [AAX_IString](#) *iEffectID1, const [AAX_IString](#) *iEffectID2)
Helper function to check if two Effect IDs are equivalent.
- [AAX_CBoolean AAX::IsAvidNotification](#) ([AAX_CTypeID](#) inNotificationID)
Helper function to check if a notification ID is reserved for host notifications.

15.286.2 Macro Definition Documentation

15.286.2.1 [_AAX_UTILSNATIVE_H_](#)

```
#define _AAX_UTILSNATIVE_H_
```

15.287 [AAX_UtilsNative.h](#)

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003
00004  * Copyright 2013-2017, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021
00022 #pragma once
00023
00024 #ifndef _AAX_UTILSNATIVE_H_
00025 #define _AAX_UTILSNATIVE_H_
00026
00027
00028 #ifndef _TMS320C6X
00029
00030 // AAX Includes
00031 #include "AAX_CString.h"
00032 #include "AAX_IString.h"
00033 #include "AAX_Assert.h"
00034 #include "AAX.h"
00035
00036 // Standard Library Includes
00037 #include <cmath> // for log()
00038 #include <string.h>
00039
00040
00041 //-----
00042 #pragma mark Utility functions
00043
00044 namespace AAX
00045 {
00046
```

```

00050     inline double SafeLog (double aValue) { return aValue <= 0.0 ? 0.0 : log(aValue); }
00051
00052     inline float SafeLogf (float aValue) { return aValue <= 0.0f ? 0.0f : logf(aValue); }
00053
00054     inline AAX_CBoolean IsParameterIDEqual ( AAX_CParamID iParam1, AAX_CParamID iParam2 ) { return
00055 static_cast<AAX_CBoolean>( strcmp ( iParam1, iParam2 ) == 0 ); }
00056
00057     inline AAX_CBoolean IsEffectIDEqual ( const AAX_IString * iEffectID1, const AAX_IString *
00058 iEffectID2 ) { return static_cast<AAX_CBoolean>( strcmp ( iEffectID1->Get(), iEffectID2->Get() ) == 0
00059 ); }
00060
00061     inline AAX_CBoolean IsAvidNotification ( AAX_CTypeID inNotificationID )
00062     {
00063         return (AAX_CBoolean)((('A' == ((inNotificationID & 0xFF000000) >> 24)) &&
00064 ('X' == ((inNotificationID & 0x00FF0000) >> 16))) ||
00065 (inNotificationID == 'ASpv'));
00066     }
00067 } // namespace AAX
00068
00069 #endif // #ifndef _TMS320C6X
00070 #endif // #ifndef _AAX_UTILSNATIVE_H_

```

15.288 AAX_VAutomationDelegate.h File Reference

```

#include "AAX_IAutomationDelegate.h"
#include "AAX_IACFAutomationDelegate.h"
#include "ACFPtr.h"

```

15.288.1 Description

Version-managed concrete AutomationDelegate class.

Classes

- class [AAX_VAutomationDelegate](#)
Version-managed concrete *automation delegate* class.

15.289 AAX_VAutomationDelegate.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00013 /*=====*/
00014
00015 #ifndef AAX_VAUTOMATIONDELEGATE_H
00016 #define AAX_VAUTOMATIONDELEGATE_H
00017
00018 #include "AAX_IAutomationDelegate.h"
00019 #include "AAX_IACFAutomationDelegate.h"
00020 #include "ACFPtr.h"

```

```

00027
00028 class AAX_IACFAutomationDelegate;
00029 class AAX_IACFController_V2;
00030 class IACFUnknown;
00031
00036 class AAX_VAutomationDelegate : public AAX_IAutomationDelegate
00037 {
00038 public:
00039     AAX_VAutomationDelegate( IACFUnknown * pUnknown );
00040     ~AAX_VAutomationDelegate() AAX_OVERRIDE;
00041
00042     IACFUnknown* GetUnknown() const { return mIAutomationDelegate; }
00043
00044     AAX_Result      RegisterParameter ( AAX_CParamID iParameterID ) AAX_OVERRIDE;
00045     AAX_Result      UnregisterParameter ( AAX_CParamID iParameterID ) AAX_OVERRIDE;
00046     AAX_Result      PostSetValueRequest ( AAX_CParamID iParameterID, double iNormalizedValue ) const
00047     AAX_OVERRIDE;
00048     AAX_Result      PostCurrentValue ( AAX_CParamID iParameterID, double iNormalizedValue ) const
00049     AAX_OVERRIDE;
00048     AAX_Result      PostTouchRequest ( AAX_CParamID iParameterID ) AAX_OVERRIDE;
00049     AAX_Result      PostReleaseRequest ( AAX_CParamID iParameterID ) AAX_OVERRIDE;
00050     AAX_Result      GetTouchState ( AAX_CParamID iParameterID, AAX_CBoolean * outTouched )
00051     AAX_OVERRIDE;
00051     AAX_Result      ParameterNameChanged ( AAX_CParamID iParameterID ) AAX_OVERRIDE;
00052
00053 private:
00054     ACFPtr<AAX_IACFAutomationDelegate> mIAutomationDelegate;
00055     ACFPtr<AAX_IACFController_V2> mIController;
00056 };
00057
00058
00059
00060 #endif //AAX_IAUTOMATIONDELEGATE_H

```

15.290 AAX_VCollection.h File Reference

```

#include "AAX.h"
#include "AAX_ICollection.h"
#include "AAX_IACFCollection.h"
#include "AAX_VDescriptionHost.h"
#include "acfunknown.h"
#include "ACFPtr.h"
#include <set>

```

15.290.1 Description

Version-managed concrete Collection class.

Classes

- class [AAX_VCollection](#)
Version-managed concrete [AAX_ICollection](#) class.

15.291 AAX_VCollection.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013–2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.

```

```

00006  *
00007  *  CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  *  not disclose to any third party. Use of the information contained in this
00009  *  document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019  /*=====*/
00020
00021 #ifndef AAX_VCOLLECTION_H
00022 #define AAX_VCOLLECTION_H
00023
00024 #include "AAX.h"
00025 #include "AAX_ICollection.h"
00026 #include "AAX_IACFCollection.h"
00027 #include "AAX_VDescriptionHost.h"
00028 #include "acfunknown.h"
00029 #include "ACFPtr.h"
00030 #include <set>
00031
00032 class IACFUnknown;
00033 class IACFPluginDefinition;
00034 class AAX_IACFCollection;
00035 class AAX_IEffectDescriptor;
00036
00041 class AAX_VCollection : public AAX_ICollection
00042 {
00043 public:
00044     AAX_VCollection (IACFUnknown * pUnkHost);
00045     ~AAX_VCollection () AAX_OVERRIDE;
00046
00051     AAX_IEffectDescriptor *      NewDescriptor () AAX_OVERRIDE;
00052     AAX_Result AddEffect ( const char * inEffectID, AAX_IEffectDescriptor *
inEffectDescriptor ) AAX_OVERRIDE;
00053     AAX_Result SetManufacturerName( const char* inPackageName ) AAX_OVERRIDE;
00054     AAX_Result AddPackageName( const char *inPackageName ) AAX_OVERRIDE;
00055     AAX_Result SetPackageVersion( uint32_t inVersion ) AAX_OVERRIDE;
00056     AAX_IPropertyMap *      NewPropertyMap () AAX_OVERRIDE;
00057     AAX_Result SetProperties ( AAX_IPropertyMap * inProperties ) AAX_OVERRIDE;
00058
00059     AAX_IDescriptionHost* DescriptionHost() AAX_OVERRIDE;
00060     const AAX_IDescriptionHost* DescriptionHost() const AAX_OVERRIDE;
00061     IACFDefinition* HostDefinition() const AAX_OVERRIDE;
00062
00063     IACFPluginDefinition*      GetIUnknown(void) const;
00064
00065 private:
00066     ACFPtr<IACFUnknown>          mUnkHost;
00067     ACFPtr<AAX_IACFCollection>   mIACFCollection;
00068     AAX_VDescriptionHost         mDescriptionHost;
00069     std::set<AAX_IEffectDescriptor *> mEffectDescriptors;
00070     std::set<AAX_IPropertyMap *>   mPropertyMaps;
00071 };
00072
00073 #endif

```

15.292 AAX_VComponentDescriptor.h File Reference

```

#include "AAX_IComponentDescriptor.h"
#include "AAX_IDma.h"
#include "AAX_IACFComponentDescriptor.h"
#include "acfunknown.h"
#include "ACFPtr.h"
#include <set>

```

15.292.1 Description

Version-managed concrete ComponentDescriptor class.

Classes

- class [AAX_VComponentDescriptor](#)
Version-managed concrete [AAX_IComponentDescriptor](#) class.

15.293 AAX_VComponentDescriptor.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #ifndef AAX_VCOMPONENTDESCRIPTOR_H
00022 #define AAX_VCOMPONENTDESCRIPTOR_H
00023
00024 // AAX Includes
00025 #include "AAX_IComponentDescriptor.h"
00026 #include "AAX_IDma.h"
00027 #include "AAX_IACFComponentDescriptor.h"
00028
00029 // ACF Includes
00030 #include "acfunknown.h"
00031 #include "ACFPtr.h"
00032
00033 // Standard Includes
00034 #include <set>
00035
00036
00037 class AAX_IPropertyMap;
00038 class AAX_IACFComponentDescriptor;
00039 class AAX_IACFComponentDescriptorV2;
00040 class IACFUnknown;
00041
00046 class AAX_VComponentDescriptor : public AAX_IComponentDescriptor
00047 {
00048 public:
00049     AAX_VComponentDescriptor ( IACFUnknown * pUnkHost );
00050     ~AAX_VComponentDescriptor () AAX_OVERRIDE;
00051
00052     AAX_Result Clear () AAX_OVERRIDE;
00053     AAX_Result AddReservedField ( AAX_CFieldIndex inFieldIndex, uint32_t inFieldType )
00054     AAX_OVERRIDE;
00055     AAX_Result AddAudioIn ( AAX_CFieldIndex inFieldIndex ) AAX_OVERRIDE;
00056     AAX_Result AddAudioOut ( AAX_CFieldIndex inFieldIndex ) AAX_OVERRIDE;
00057     AAX_Result AddAudioBufferLength ( AAX_CFieldIndex inFieldIndex ) AAX_OVERRIDE;
00058     AAX_Result AddSampleRate ( AAX_CFieldIndex inFieldIndex ) AAX_OVERRIDE;
00059     AAX_Result AddClock ( AAX_CFieldIndex inFieldIndex ) AAX_OVERRIDE;
00060     AAX_Result AddSideChainIn ( AAX_CFieldIndex inFieldIndex ) AAX_OVERRIDE;
00061     AAX_Result AddDataInPort ( AAX_CFieldIndex inFieldIndex, uint32_t inPacketSize,
00062     AAX_EDataInPortType inPortType ) AAX_OVERRIDE;
00063     AAX_Result AddAuxOutputStem ( AAX_CFieldIndex inFieldIndex, int32_t inStemFormat, const
00064     char inNameUTF8[]) AAX_OVERRIDE;
00065     AAX_Result AddPrivateData ( AAX_CFieldIndex inFieldIndex, int32_t inDataSize, uint32_t
00066     inOptions ) AAX_OVERRIDE;
00067     AAX_Result AddTemporaryData( AAX_CFieldIndex inFieldIndex, uint32_t inDataElementSize)
00068     AAX_OVERRIDE;
00069     AAX_Result AddDmaInstance ( AAX_CFieldIndex inFieldIndex, AAX_IDma::EMode inDmaMode )
00070     AAX_OVERRIDE;
00071     AAX_Result AddMeters ( AAX_CFieldIndex inFieldIndex, const AAX_CTypeID* inMeterIDs,
00072     const uint32_t inMeterCount) AAX_OVERRIDE;
00073     AAX_Result AddMIDINode ( AAX_CFieldIndex inFieldIndex, AAX_EMIDINodeType inNodeType,
00074     const char inNodeName[], uint32_t channelMask ) AAX_OVERRIDE;
00075
00076     AAX_IPropertyMap * NewPropertyMap () const AAX_OVERRIDE;
00077     AAX_IPropertyMap * DuplicatePropertyMap (AAX_IPropertyMap* inPropertyMap) const AAX_OVERRIDE;
00078     virtual AAX_Result AddProcessProc_Native (
00079     AAX_CProcessProc inProcessProc,

```

```

00084         AAX_IPropertyMap * inProperties = NULL,
00085         AAX_CInstanceInitProc inInstanceInitProc = NULL,
00086         AAX_CBackgroundProc inBackgroundProc = NULL,
00087         AAX_CSelector * outProcID = NULL ) AAX_OVERRIDE;
00090     virtual AAX_Result      AddProcessProc_TI (
00091         const char inDLLFileNameUTF8[],
00092         const char inProcessProcSymbol[],
00093         AAX_IPropertyMap * inProperties = NULL,
00094         const char inInstanceInitProcSymbol [] = NULL,
00095         const char inBackgroundProcSymbol [] = NULL,
00096         AAX_CSelector * outProcID = NULL ) AAX_OVERRIDE;
00099     virtual AAX_Result AddProcessProc (
00100         AAX_IPropertyMap* inProperties,
00101         AAX_CSelector* outProcIDs = NULL,
00102         int32_t inProcIDsSize = 0) AAX_OVERRIDE;
00103
00104
00105     IACFUnknown*      GetIUnknown(void) const;
00106
00107 private:
00108     // Used for backwards compatibility with clients which do not support AddProcessProc
00109     friend class AAX_VPropertyMap;
00110     static const std::set<AAX_EProperty>& PointerPropertiesUsedByAddProcessProc();
00111
00112 private:
00113     ACFPtr<IACFUnknown> mUnkHost;
00114     ACFPtr<AAX_IACFComponentDescriptor> mIACFComponentDescriptor;
00115     ACFPtr<AAX_IACFComponentDescriptor_V2> mIACFComponentDescriptorV2;
00116     ACFPtr<AAX_IACFComponentDescriptor_V3> mIACFComponentDescriptorV3;
00117     std::set<AAX_IPropertyMap *> mPropertyMaps;
00118 };
00119
00120
00121 #endif // #ifndef _AAX_ICOMPONENTDESCRIPTOR_H_

```

15.294 AAX_VController.h File Reference

```

#include "AAX_IController.h"
#include "AAX_IACFController.h"
#include "ACFPtr.h"

```

15.294.1 Description

Version-managed concrete Controller class.

Classes

- class [AAX_VController](#)
Version-managed concrete *Controller* class.

15.295 AAX_VController.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *

```

```

00011  */
00012
00019  /*=====*/
00020
00021  #ifndef AAX_VCONTROLLER_H
00022  #define AAX_VCONTROLLER_H
00023
00024  #include "AAX_IController.h"
00025  #include "AAX_IACFController.h"
00026
00027  #ifdef __clang__
00028  #pragma clang diagnostic push
00029  #pragma clang diagnostic ignored "-Wself-assign"
00030  #pragma clang diagnostic ignored "-Wnon-virtual-dtor"
00031  #endif
00032
00033  #include "ACFPtr.h"
00034
00035  #ifdef __clang__
00036  #pragma clang diagnostic pop
00037  #endif
00038
00039  class IACFUnknown;
00040  class IACFComponentFactory;
00041  class AAX_IACFPageTableController;
00042  class AAX_IACFPageTableController_V2;
00043  class AAX_IACFPageTable_V2;
00044
00052  class AAX_VController : public AAX_IController
00053  {
00054  public:
00055      AAX_VController( IACFUnknown* pUnknown );
00056      ~AAX_VController() override;
00057
00058      //Host Information Getters
00059      AAX_Result      GetEffectID ( AAX_IString *   outEffectID) const AAX_OVERRIDE;
00060      AAX_Result      GetSampleRate ( AAX_CSampleRate * outSampleRate ) const AAX_OVERRIDE;
00061      AAX_Result      GetInputStemFormat ( AAX_EStemFormat * outStemFormat ) const AAX_OVERRIDE;
00062      AAX_Result      GetOutputStemFormat ( AAX_EStemFormat * outStemFormat ) const AAX_OVERRIDE;
00063      AAX_Result      GetSignalLatency( int32_t* outSamples) const AAX_OVERRIDE;
00064      AAX_Result      GetHybridSignalLatency(int32_t* outSamples) const AAX_OVERRIDE;
00065      AAX_Result      GetPlugInTargetPlatform(AAX_CTargetPlatform* outTargetPlatform) const AAX_OVERRIDE;
00066      AAX_Result      GetIsAudioSuite(AAX_CBoolean* outIsAudioSuite) const AAX_OVERRIDE;
00067      AAX_Result      GetCycleCount( AAX_EProperty inWhichCycleCount, AAX_CPropertyValue* outNumCycles)
00068      const AAX_OVERRIDE;
00069      AAX_Result      GetTODLocation ( AAX_CTimeOfDay* outTODLocation ) const AAX_OVERRIDE;
00069      AAX_Result      GetCurrentAutomationTimestamp(AAX_CTransportCounter* outTimestamp) const
00070      AAX_OVERRIDE;
00070      AAX_Result      GetHostName(AAX_IString* outHostNameString) const AAX_OVERRIDE;
00071
00072      //Host Information Setters (Dynamic info)
00073      AAX_Result      SetSignalLatency(int32_t inNumSamples) AAX_OVERRIDE;
00074      AAX_Result      SetCycleCount( AAX_EProperty* inWhichCycleCounts, AAX_CPropertyValue* iValues,
00075      int32_t numValues) AAX_OVERRIDE;
00076
00076      //Posting functions.
00077      AAX_Result      PostPacket ( AAX_CFieldIndex inFieldIndex, const void * inPayloadP, uint32_t
00078      inPayloadSize ) AAX_OVERRIDE;
00079
00079      // Notification functions
00080      AAX_Result      SendNotification ( AAX_CTypeID inNotificationType, const void* inNotificationData,
00081      uint32_t inNotificationDataSize ) AAX_OVERRIDE;
00081      AAX_Result      SendNotification( AAX_CTypeID inNotificationType) AAX_OVERRIDE;
00082
00083      //Metering functions
00084      AAX_Result      GetCurrentMeterValue ( AAX_CTypeID inMeterID, float * outMeterValue ) const
00085      AAX_OVERRIDE;
00085      AAX_Result      GetMeterPeakValue( AAX_CTypeID inMeterID, float * outMeterPeakValue ) const
00086      AAX_OVERRIDE;
00086      AAX_Result      ClearMeterPeakValue ( AAX_CTypeID inMeterID ) const AAX_OVERRIDE;
00087      AAX_Result      GetMeterClipped ( AAX_CTypeID inMeterID, AAX_CBoolean * outClipped ) const
00088      AAX_OVERRIDE;
00088      AAX_Result      ClearMeterClipped ( AAX_CTypeID inMeterID ) const AAX_OVERRIDE;
00089      AAX_Result      GetMeterCount ( uint32_t * outMeterCount ) const AAX_OVERRIDE;
00090
00091      //MIDI functions
00092      AAX_Result      GetNextMIDIPacket( AAX_CFieldIndex* outPort, AAX_CMidiPacket* outPacket )
00093      AAX_OVERRIDE;
00094
00094      // PageTables functions
00097      AAX_IPageTable*
00098      CreateTableCopyForEffect(AAX_CPropertyValue inManufacturerID,
00099      AAX_CPropertyValue inProductID,
00100      AAX_CPropertyValue inPlugInID,
00101      uint32_t inTableType,
00102      int32_t inTablePageSize) const AAX_OVERRIDE;
00105      AAX_IPageTable*

```



```

00106     CreateTableCopyForLayout(const char * inEffectID,
00107                             const char * inLayoutName,
00108                             uint32_t inTableType,
00109                             int32_t inTablePageSize) const AAX_OVERRIDE;
00112     AAX_IPageTable*
00113     CreateTableCopyForEffectFromFile(const char* inPageTableFilePath,
00114                                     AAX_ETextEncoding inFilePathEncoding,
00115                                     AAX_CPropertyValue inManufacturerID,
00116                                     AAX_CPropertyValue inProductID,
00117                                     AAX_CPropertyValue inPlugInID,
00118                                     uint32_t inTableType,
00119                                     int32_t inTablePageSize) const AAX_OVERRIDE;
00122     AAX_IPageTable*
00123     CreateTableCopyForLayoutFromFile(const char* inPageTableFilePath,
00124                                     AAX_ETextEncoding inFilePathEncoding,
00125                                     const char* inLayoutName,
00126                                     uint32_t inTableType,
00127                                     int32_t inTablePageSize) const AAX_OVERRIDE;
00128 private:
00129     ACFPtr<AAX_IACFPageTable_V2> CreatePageTable() const;
00140 private:
00142     ACFPtr<AAX_IACFController>      mIController;
00143     ACFPtr<AAX_IACFController_V2>  mIControllerV2;
00144     ACFPtr<AAX_IACFController_V3>  mIControllerV3;
00145     // AAX_IACFPageTableController interface methods are aggregated into AAX_IController
00147     ACFPtr<AAX_IACFPageTableController>      mIPageTableController;
00148     ACFPtr<AAX_IACFPageTableController_V2>  mIPageTableControllerV2;
00150     ACFPtr<IACFComponentFactory>      mComponentFactory;
00151 };
00152
00153 #endif // AAX_VCONTROLLER_H
00155

```

15.296 AAX_VDataBufferWrapper.h File Reference

```

#include "AAX_IDataBufferWrapper.h"
#include "ACFPtr.h"

```

Classes

- class [AAX_VDataBufferWrapper](#)
Wrapper for an [AAX_IDataBuffer](#).

Macros

- #define [AAX_VDATABUFFERWRAPPER_H](#)

15.296.1 Macro Definition Documentation

15.296.1.1 AAX_VDATABUFFERWRAPPER_H

```
#define AAX_VDATABUFFERWRAPPER_H
```

15.297 AAX_VDataBufferWrapper.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019 #ifndef AAX_VDATABUFFERWRAPPER_H
00020 #define AAX_VDATABUFFERWRAPPER_H
00021
00022 #include "AAX_IDataBufferWrapper.h"
00023 #include "ACFPtr.h"
00024
00025 class IACFUnknown;
00026 class AAX_IACFDataBuffer;
00027
00039 class AAX_VDataBufferWrapper : public AAX_IDataBufferWrapper
00040 {
00041 public:
00042     explicit AAX_VDataBufferWrapper(IACFUnknown * iUnknown);
00043     ~AAX_VDataBufferWrapper() AAX_OVERRIDE;
00044
00045     AAX_Result Type(AAX_CTypeID * oType) const AAX_OVERRIDE;
00046     AAX_Result Size(int32_t * oSize) const AAX_OVERRIDE;
00047     AAX_Result Data(void const ** oBuffer) const AAX_OVERRIDE;
00048
00049 private:
00050     ACFPtr<AAX_IACFDataBuffer> mDataBufferV1;
00051 };
00052
00053 #endif // AAX_VDATABUFFERWRAPPER_H

```

15.298 AAX_VDescriptionHost.h File Reference

```

#include "AAX_IDescriptionHost.h"
#include "ACFPtr.h"

```

Classes

- class [AAX_VDescriptionHost](#)

15.299 AAX_VDescriptionHost.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2019, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  *
00010
00011 #ifndef AAXLibrary_AAX_VDescriptionHost_h
00012 #define AAXLibrary_AAX_VDescriptionHost_h

```

```

00013
00014
00015 #include "AAX_IDescriptionHost.h"
00016 #include "ACFPtr.h"
00017
00018
00019 class AAX_IACFDescriptionHost;
00020 class IACFDefinition;
00021
00022
00029 class AAX_VDescriptionHost : public AAX_IDescriptionHost
00030 {
00031 public:
00032     explicit AAX_VDescriptionHost( IACFUnknown* pUnknown );
00033     ~AAX_VDescriptionHost() AAX_OVERRIDE;
00034
00035 public: // AAX_IDescriptionHost
00036     const AAX_IFeatureInfo* AcquireFeatureProperties(const AAX_Feature_UID& inFeatureID) const
00037         AAX_OVERRIDE;
00038
00038 public: // AAX_VDescriptionHost
00039     bool Supported() const { return !mDescriptionHost.isNull(); }
00040     AAX_IACFDescriptionHost* DescriptionHost() { return mDescriptionHost.inArg(); } // does not addrf
00041     const AAX_IACFDescriptionHost* DescriptionHost() const { return mDescriptionHost.inArg(); } //
00042     does not addrf
00043     IACFDefinition* HostDefinition() const { return mHostInformation.inArg(); } // does not addrf
00044
00044 private:
00045     ACFPtr<AAX_IACFDescriptionHost> mDescriptionHost;
00046     ACFPtr<IACFDefinition> mHostInformation;
00047 };
00048
00049
00050
00051
00052 #endif // AAXLibrary_AAX_VDescriptionHost_h

```

15.300 AAX_VEffectDescriptor.h File Reference

```

#include "AAX.h"
#include "AAX_IEffectDescriptor.h"
#include "AAX_IACFEffectDescriptor.h"
#include "acfunknown.h"
#include "ACFPtr.h"
#include <set>
#include <map>

```

15.300.1 Description

Version-managed concrete EffectDescriptor class.

Classes

- class [AAX_VEffectDescriptor](#)
Version-managed concrete [AAX_IEffectDescriptor](#) class.

15.301 AAX_VEffectDescriptor.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #ifndef AAX_VEFFECTDESCRIPTOR_H
00022 #define AAX_VEFFECTDESCRIPTOR_H
00023
00024 #include "AAX.h"
00025 #include "AAX_IEffectDescriptor.h"
00026 #include "AAX_IACFEEffectDescriptor.h"
00027 #include "acfunknown.h"
00028 #include "ACFPtr.h"
00029
00030 #include <set>
00031 #include <map>
00032
00033 class AAX_IComponentDescriptor;
00034 class AAX_IPropertyMap;
00035 class AAX_IACFEEffectDescriptor;
00036 class IACFUnknown;
00037
00042 class AAX_VEffectDescriptor : public AAX_IEffectDescriptor
00043 {
00044 public:
00045     AAX_VEffectDescriptor ( IACFUnknown * pUnkHost );
00046     ~AAX_VEffectDescriptor () AAX_OVERRIDE;
00047
00052     AAX_IComponentDescriptor *      NewComponentDescriptor () AAX_OVERRIDE;
00053     AAX_Result                      AddComponent ( AAX_IComponentDescriptor * inComponentDescriptor
00054 ) AAX_OVERRIDE;
00054     AAX_Result                      AddName ( const char * inPlugInName ) AAX_OVERRIDE;
00055     AAX_Result                      AddCategory ( uint32_t inCategory ) AAX_OVERRIDE;
00056     AAX_Result                      AddCategoryBypassParameter ( uint32_t inCategory, AAX_CParamID
inParamID ) AAX_OVERRIDE;
00057     AAX_Result                      AddProcPtr ( void * inProcPtr, AAX_CProcPtrID inProcID )
AAX_OVERRIDE;
00062     AAX_IPropertyMap *              NewPropertyMap () AAX_OVERRIDE;
00063     AAX_Result                      SetProperties ( AAX_IPropertyMap * inProperties ) AAX_OVERRIDE;
00064     AAX_Result                      AddResourceInfo ( AAX_EResourceType inResourceType, const char *
inInfo ) AAX_OVERRIDE;
00065     AAX_Result                      AddMeterDescription( AAX_CTypeID inMeterID, const char *
inMeterName, AAX_IPropertyMap * inProperties ) AAX_OVERRIDE;
00066     AAX_Result                      AddControlMIDINode ( AAX_CTypeID inNodeID, AAX_EMIDINodeType
inNodeType, const char inNodeName[], uint32_t inChannelMask ) AAX_OVERRIDE;
00067
00068     IACFUnknown*                   GetIUnknown(void) const;
00069
00070 private:
00071     ACFPtr<IACFUnknown>              mUnkHost;
00072     ACFPtr<AAX_IACFEEffectDescriptor> mIACFEEffectDescriptor;
00073     ACFPtr<AAX_IACFEEffectDescriptor_V2> mIACFEEffectDescriptorV2;
00074     std::set<AAX_IComponentDescriptor *> mComponentDescriptors;
00075     std::set<AAX_IPropertyMap *>        mPropertyMaps;
00076
00077 };
00078
00079 #endif // AAX_VEFFECTDESCRIPTOR_H

```

15.302 AAX_Version.h File Reference

15.302.1 Description

Version stamp header for the AAX SDK.

This file defines a unique number that can be used to identify the version of the AAX SDK

Macros

- `#define _AAX_VERSION_H_`
- `#define AAX_SDK_VERSION (0x0206)`
The SDK's version number.
- `#define AAX_SDK_CURRENT_REVISION (20206010)`
An atomic revision number for the source included in this SDK.
- `#define AAX_SDK_1p0p1_REVISION (3712639)`
- `#define AAX_SDK_1p0p2_REVISION (3780585)`
- `#define AAX_SDK_1p0p3_REVISION (3895859)`
- `#define AAX_SDK_1p0p4_REVISION (4333589)`
- `#define AAX_SDK_1p0p5_REVISION (4598560)`
- `#define AAX_SDK_1p0p6_REVISION (5051497)`
- `#define AAX_SDK_1p5p0_REVISION (5740047)`
- `#define AAX_SDK_2p0b1_REVISION (6169787)`
- `#define AAX_SDK_2p0p0_REVISION (6307708)`
- `#define AAX_SDK_2p0p1_REVISION (6361692)`
- `#define AAX_SDK_2p1p0_REVISION (7820991)`
- `#define AAX_SDK_2p1p1_REVISION (8086416)`
- `#define AAX_SDK_2p2p0_REVISION (9967334)`
- `#define AAX_SDK_2p2p1_REVISION (10693954)`
- `#define AAX_SDK_2p2p2_REVISION (11819832)`
- `#define AAX_SDK_2p3p0_REVISION (12546840)`
- `#define AAX_SDK_2p3p1_REVISION (13200373)`
- `#define AAX_SDK_2p3p2_REVISION (14017972)`
- `#define AAX_SDK_2p4p0_REVISION (20204000)`
- `#define AAX_SDK_2p4p1_REVISION (20204010)`
- `#define AAX_SDK_2p5p0_REVISION (20205000)`
- `#define AAX_SDK_2p6p0_REVISION (20206000)`
- `#define AAX_SDK_2p6p1_REVISION (20206001)`

15.302.2 Macro Definition Documentation

15.302.2.1 _AAX_VERSION_H_

```
#define _AAX_VERSION_H_
```

15.302.2.2 AAX_SDK_VERSION

```
#define AAX_SDK_VERSION ( 0x0206 )
```

The SDK's version number.

This version number is generally updated only when changes have been made to the AAX binary interface

- The first byte is the major version number
- The second byte is the minor version number

For example:

- SDK 1.0.5 > 0x0100
- SDK 10.2.1 > 0x0A02

15.302.2.3 AAX_SDK_CURRENT_REVISION

```
#define AAX_SDK_CURRENT_REVISION ( 20206010 )
```

An atomic revision number for the source included in this SDK.

15.302.2.4 AAX_SDK_1p0p1_REVISION

```
#define AAX_SDK_1p0p1_REVISION ( 3712639 )
```

15.302.2.5 AAX_SDK_1p0p2_REVISION

```
#define AAX_SDK_1p0p2_REVISION ( 3780585 )
```

15.302.2.6 AAX_SDK_1p0p3_REVISION

```
#define AAX_SDK_1p0p3_REVISION ( 3895859 )
```

15.302.2.7 AAX_SDK_1p0p4_REVISION

```
#define AAX_SDK_1p0p4_REVISION ( 4333589 )
```

15.302.2.8 AAX_SDK_1p0p5_REVISION

```
#define AAX_SDK_1p0p5_REVISION ( 4598560 )
```

15.302.2.9 AAX_SDK_1p0p6_REVISION

```
#define AAX_SDK_1p0p6_REVISION ( 5051497 )
```

15.302.2.10 AAX_SDK_1p5p0_REVISION

```
#define AAX_SDK_1p5p0_REVISION ( 5740047 )
```

15.302.2.11 AAX_SDK_2p0b1_REVISION

```
#define AAX_SDK_2p0b1_REVISION ( 6169787 )
```

15.302.2.12 AAX_SDK_2p0p0_REVISION

```
#define AAX_SDK_2p0p0_REVISION ( 6307708 )
```

15.302.2.13 AAX_SDK_2p0p1_REVISION

```
#define AAX_SDK_2p0p1_REVISION ( 6361692 )
```

15.302.2.14 AAX_SDK_2p1p0_REVISION

```
#define AAX_SDK_2p1p0_REVISION ( 7820991 )
```

15.302.2.15 AAX_SDK_2p1p1_REVISION

```
#define AAX_SDK_2p1p1_REVISION ( 8086416 )
```

15.302.2.16 AAX_SDK_2p2p0_REVISION

```
#define AAX_SDK_2p2p0_REVISION ( 9967334 )
```

15.302.2.17 AAX_SDK_2p2p1_REVISION

```
#define AAX_SDK_2p2p1_REVISION ( 10693954 )
```

15.302.2.18 AAX_SDK_2p2p2_REVISION

```
#define AAX_SDK_2p2p2_REVISION ( 11819832 )
```

15.302.2.19 AAX_SDK_2p3p0_REVISION

```
#define AAX_SDK_2p3p0_REVISION ( 12546840 )
```

15.302.2.20 AAX_SDK_2p3p1_REVISION

```
#define AAX_SDK_2p3p1_REVISION ( 13200373 )
```

15.302.2.21 AAX_SDK_2p3p2_REVISION

```
#define AAX_SDK_2p3p2_REVISION ( 14017972 )
```

15.302.2.22 AAX_SDK_2p4p0_REVISION

```
#define AAX_SDK_2p4p0_REVISION ( 20204000 )
```

15.302.2.23 AAX_SDK_2p4p1_REVISION

```
#define AAX_SDK_2p4p1_REVISION ( 20204010 )
```

15.302.2.24 AAX_SDK_2p5p0_REVISION

```
#define AAX_SDK_2p5p0_REVISION ( 20205000 )
```

15.302.2.25 AAX_SDK_2p6p0_REVISION

```
#define AAX_SDK_2p6p0_REVISION ( 20206000 )
```


15.302.2.26 AAX_SDK_2p6p1_REVISION

```
#define AAX_SDK_2p6p1_REVISION ( 20206001 )
```

15.303 AAX_Version.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003
00004  * Copyright 2013-2017, 2019, 2021-2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011 */
00012
00020 /*=====*/
00021
00022
00023 #pragma once
00024
00025 #ifndef _AAX_VERSION_H_
00026 #define _AAX_VERSION_H_
00027
00028
00043 #define AAX_SDK_VERSION ( 0x0206 )
00044
00047 #define AAX_SDK_CURRENT_REVISION ( 20206010 )
00048
00049
00050 #define AAX_SDK_1p0p1_REVISION ( 3712639 )
00051 #define AAX_SDK_1p0p2_REVISION ( 3780585 )
00052 #define AAX_SDK_1p0p3_REVISION ( 3895859 )
00053 #define AAX_SDK_1p0p4_REVISION ( 4333589 )
00054 #define AAX_SDK_1p0p5_REVISION ( 4598560 )
00055 #define AAX_SDK_1p0p6_REVISION ( 5051497 )
00056 #define AAX_SDK_1p5p0_REVISION ( 5740047 )
00057 #define AAX_SDK_2p0b1_REVISION ( 6169787 )
00058 #define AAX_SDK_2p0p0_REVISION ( 6307708 )
00059 #define AAX_SDK_2p0p1_REVISION ( 6361692 )
00060 #define AAX_SDK_2p1p0_REVISION ( 7820991 )
00061 #define AAX_SDK_2p1p1_REVISION ( 8086416 )
00062 #define AAX_SDK_2p2p0_REVISION ( 9967334 )
00063 #define AAX_SDK_2p2p1_REVISION ( 10693954 )
00064 #define AAX_SDK_2p2p2_REVISION ( 11819832 )
00065 #define AAX_SDK_2p3p0_REVISION ( 12546840 )
00066 #define AAX_SDK_2p3p1_REVISION ( 13200373 )
00067 #define AAX_SDK_2p3p2_REVISION ( 14017972 )
00068 #define AAX_SDK_2p4p0_REVISION ( 20204000 )
00069 #define AAX_SDK_2p4p1_REVISION ( 20204010 )
00070 #define AAX_SDK_2p5p0_REVISION ( 20205000 )
00071 #define AAX_SDK_2p6p0_REVISION ( 20206000 )
00072 #define AAX_SDK_2p6p1_REVISION ( 20206001 )
00073 //CURREVSTAMP < do not remove this comment
00074
00075
00076
00077 #endif // #ifndef _AAX_VERSION_H_
```

15.304 AAX_VFeatureInfo.h File Reference

```
#include "AAX_IFeatureInfo.h"
#include "ACFPtr.h"
#include "acfbasetypes.h"
```

Classes

- class [AAX_VFeatureInfo](#)

15.305 AAX_VFeatureInfo.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2016-2017, 2019, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00011 #ifndef AAXLibrary_AAX_VFeatureInfo_h
00012 #define AAXLibrary_AAX_VFeatureInfo_h
00013
00014 #include "AAX_IFeatureInfo.h"
00015
00016 #include "ACFPtr.h"
00017 #include "acfbasetypes.h"
00018
00019
00020 class AAX_IPropertyMap;
00021 class AAX_IACFFeatureInfo;
00022
00023
00027 class AAX_VFeatureInfo : public AAX_IFeatureInfo
00028 {
00029 public:
00030     explicit AAX_VFeatureInfo( IACFUnknown* pUnknown, const AAX_Feature_UID& inFeatureID );
00031     ~AAX_VFeatureInfo() AAX_OVERRIDE;
00032
00033 public: // AAX_IFeatureInfo
00034     AAX_Result SupportLevel( AAX_ESupportLevel& oSupportLevel) const AAX_OVERRIDE;
00035     const AAX_IPropertyMap* AcquireProperties() const AAX_OVERRIDE;
00036     const AAX_Feature_UID& ID() const AAX_OVERRIDE;
00037
00038 private:
00039     AAX_Feature_UID mFeatureID;
00040     ACFPtr<AAX_IACFFeatureInfo> mIFeature;
00041 };
00042
00043
00044 #endif
```

15.306 AAX_VHostProcessorDelegate.h File Reference

```
#include "AAX_IHostProcessorDelegate.h"
#include "AAX_IACFHostProcessorDelegate.h"
#include "ACFPtr.h"
```

15.306.1 Description

Version-managed concrete HostProcessorDelegate class.

Classes

- class [AAX_VHostProcessorDelegate](#)
Version-managed concrete *Host Processor delegate* class.

15.307 AAX_VHostProcessorDelegate.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #ifndef AAX_VHOSTPROCESSORDELEGATE_H
00022 #define AAX_VHOSTPROCESSORDELEGATE_H
00023
00024 #include "AAX_IHostProcessorDelegate.h"
00025 #include "AAX_IACFHostProcessorDelegate.h"
00026 #include "ACFPtr.h"
00027
00028
00029 class IACFUnknown;
00030 class AAX_IACFHostProcessorDelegate;
00031
00037 class AAX_VHostProcessorDelegate : public AAX_IHostProcessorDelegate
00038 {
00039 public:
00040     AAX_VHostProcessorDelegate( IACFUnknown* pUnknown );
00041
00042     AAX_Result      GetAudio ( const float * const inAudioIns [], int32_t inAudioInCount, int64_t
inLocation, int32_t * ioNumSamples ) AAX_OVERRIDE;
00043     int32_t         GetSideChainInputNum () AAX_OVERRIDE;
00044     AAX_Result      ForceAnalyze () AAX_OVERRIDE;
00045     AAX_Result      ForceProcess () AAX_OVERRIDE;
00046
00047 private:
00048     ACFPtr<AAX_IACFHostProcessorDelegate>      mIHostProcessorDelegate;
00049     ACFPtr<AAX_IACFHostProcessorDelegate_V2>    mIHostProcessorDelegateV2;
00050     ACFPtr<AAX_IACFHostProcessorDelegate_V3>    mIHostProcessorDelegateV3;
00051 };
00052
00053
00054
00055 #endif //AAX_IAUTOMATIONDELEGATE_H
00056
```

15.308 AAX_VHostServices.h File Reference

```
#include "AAX_IHostServices.h"
#include "AAX.h"
#include "acfunknown.h"
#include "ACFPtr.h"
#include "AAX_IACFHostServices.h"
```

15.308.1 Description

Version-managed concrete HostServices class.

Classes

- class [AAX_VHostServices](#)
Version-managed concrete [AAX_IHostServices](#) class.

15.309 AAX_VHostServices.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   *
00004   * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005   * All rights reserved.
00006   *
00007   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008   * not disclose to any third party. Use of the information contained in this
00009   * document is subject to an Avid SDK license.
00010   *
00011   */
00012
00013  /*=====*/
00014  #ifndef AAX_VHOSTSERVICES_H
00015  #define AAX_VHOSTSERVICES_H
00016
00017  #include "AAX_IHostServices.h"
00018  #include "AAX.h"
00019  #include "acfunknown.h"
00020  #include "ACFPtr.h"
00021  #include "AAX_IACFHostServices.h"
00022
00023
00024
00025  class IACFUnknown;
00026  class AAX_IACFHostServices;
00027
00028  class AAX_VHostServices : public AAX_IHostServices
00029  {
00030  public:
00031      AAX_VHostServices( IACFUnknown * pUnkHost );
00032      ~AAX_VHostServices( );
00033
00034      AAX_Result HandleAssertFailure ( const char * iFile, int32_t iLine, const char * iNote, /*
00035      AAX_EAssertFlags */ int32_t iFlags ) const AAX_OVERRIDE;
00036      AAX_Result Trace ( int32_t iPriority, const char * iMessage ) const AAX_OVERRIDE;
00037      AAX_Result StackTrace ( int32_t iTracePriority, int32_t iStackTracePriority, const char * iMessage
00038      ) const AAX_OVERRIDE;
00039
00040  private:
00041      ACFPtr<AAX_IACFHostServices> mIACFHostServices;
00042      ACFPtr<AAX_IACFHostServices_V2> mIACFHostServices2;
00043      ACFPtr<AAX_IACFHostServices_V3> mIACFHostServices3;
00044  };
00045
00046
00047
00048
00049  #endif //AAX_IAUTOMATIONDELEGATE_H
00050
00051
00052
00053
00054
00055
00056
00057
00058

```

15.310 AAX_VPageTable.h File Reference

```

#include "AAX_IPageTable.h"
#include "AAX_IACFPageTable.h"
#include "ACFPtr.h"

```

Classes

- class [AAX_VPageTable](#)
Version-managed concrete [AAX_IPageTable](#) class.

15.311 AAX_VPageTable.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   * Copyright 2016-2017, 2019, 2023 Avid Technology, Inc.
00004   * All rights reserved.
00005   *
00006   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007   * not disclose to any third party. Use of the information contained in this
00008   * document is subject to an Avid SDK license.
00009   */
00010
00011 #ifndef AAXLibrary_AAX_VPageTable_h
00012 #define AAXLibrary_AAX_VPageTable_h
00013
00014 #include "AAX_IPageTable.h"
00015 #include "AAX_IACFPPageTable.h"
00016 #include "ACFPtr.h"
00017
00022 class AAX_VPageTable : public AAX_IPageTable
00023 {
00024 public:
00025     AAX_VPageTable( IACFUnknown* pUnknown );
00026     ~AAX_VPageTable() AAX_OVERRIDE;
00027
00028     // AAX_IACFPPageTable
00029     AAX_Result Clear() AAX_OVERRIDE;
00030     AAX_Result Empty(AAX_CBoolean& oEmpty) const AAX_OVERRIDE;
00031     AAX_Result GetNumPages(int32_t& oNumPages) const AAX_OVERRIDE;
00032     AAX_Result InsertPage(int32_t iPage) AAX_OVERRIDE;
00033     AAX_Result RemovePage(int32_t iPage) AAX_OVERRIDE;
00034     AAX_Result GetNumMappedParameterIDs(int32_t iPage, int32_t& oNumParameterIdentifiers) const
00035     AAX_OVERRIDE;
00036     AAX_Result ClearMappedParameter(int32_t iPage, int32_t iIndex) AAX_OVERRIDE;
00037     AAX_Result GetMappedParameterID(int32_t iPage, int32_t iIndex, AAX_IString& oParameterIdentifier)
00038     const AAX_OVERRIDE;
00039     AAX_Result MapParameterID(AAX_CParamID iParameterIdentifier, int32_t iPage, int32_t iIndex)
00040     AAX_OVERRIDE;
00041
00042     // AAX_IACFPPageTable_V2
00043     AAX_Result GetNumParametersWithNameVariations(int32_t& oNumParameterIdentifiers) const
00044     AAX_OVERRIDE;
00045     AAX_Result GetNameVariationParameterIDAtIndex(int32_t iIndex, AAX_IString& oParameterIdentifier)
00046     const AAX_OVERRIDE;
00047     AAX_Result GetNumNameVariationsForParameter(AAX_CPageTableParamID iParameterIdentifier, int32_t&
00048     oNumVariations) const AAX_OVERRIDE;
00049     AAX_Result GetParameterNameVariationAtIndex(AAX_CPageTableParamID iParameterIdentifier, int32_t
00050     iIndex, AAX_IString& oNameVariation, int32_t& oLength) const AAX_OVERRIDE;
00051     AAX_Result GetParameterNameVariationOfLength(AAX_CPageTableParamID iParameterIdentifier, int32_t
00052     iLength, AAX_IString& oNameVariation) const AAX_OVERRIDE;
00053     AAX_Result ClearParameterNameVariations() AAX_OVERRIDE;
00054     AAX_Result ClearNameVariationsForParameter(AAX_CPageTableParamID iParameterIdentifier)
00055     AAX_OVERRIDE;
00056     AAX_Result SetParameterNameVariation(AAX_CPageTableParamID iParameterIdentifier, const
00057     AAX_IString& iNameVariation, int32_t iLength) AAX_OVERRIDE;
00058
00059     // AAX_VPageTable
00060     const IACFUnknown* AsUnknown() const
00061     {
00062         return mIPageTable.inArg();
00063     }
00064     IACFUnknown* AsUnknown()
00065     {
00066         return mIPageTable.inArg();
00067     }
00068     bool IsSupported() const { return !mIPageTable.isNull(); }
00069
00070 private:
00071     ACFPtr<AAX_IACFPPageTable> mIPageTable;
00072     ACFPtr<AAX_IACFPPageTable_V2> mIPageTable2;
00073 };
00074 #endif

```

15.312 AAX_VPrivateDataAccess.h File Reference

```
#include "AAX_IPrivateDataAccess.h"
#include "AAX_IACFPrivateDataAccess.h"
#include "ACFPtr.h"
```

15.312.1 Description

Version-managed concrete PrivateDataAccess class.

Classes

- class [AAX_VPrivateDataAccess](#)
Version-managed concrete [AAX_IPrivateDataAccess](#) class.

15.313 AAX_VPrivateDataAccess.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2014-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012 /*=====*/
00019 /*=====*/
00020 #ifndef AAX_VPRIVATEDATAACCESS_H
00021 #define AAX_VPRIVATEDATAACCESS_H
00022
00023 #include "AAX_IPrivateDataAccess.h"
00024 #include "AAX_IACFPrivateDataAccess.h"
00025 #include "ACFPtr.h"
00026
00027
00028 class IACFUnknown;
00029
00034 class AAX_VPrivateDataAccess : public AAX_IPrivateDataAccess
00035 {
00036 public:
00037     AAX_VPrivateDataAccess( IACFUnknown* pUnknown );
00038     ~AAX_VPrivateDataAccess() AAX_OVERRIDE;
00039
00040     // Direct access methods
00041     AAX_Result ReadPortDirect( AAX_CFieldIndex inFieldIndex, const uint32_t inOffset, const
uint32_t inSize, void* outBuffer ) AAX_OVERRIDE;
00042     AAX_Result WritePortDirect( AAX_CFieldIndex inFieldIndex, const uint32_t inOffset, const
uint32_t inSize, const void* inBuffer ) AAX_OVERRIDE;
00043
00044 private:
00045     AAX_IACFPrivateDataAccess* mIPrivateDataAccess;
00046 };
00047
00048
00049
00050 #endif //AAX_VPRIVATEDATAACCESS_H
00051
```

15.314 AAX_VPropertyMap.h File Reference

```
#include "AAX_IPropertyMap.h"
#include "AAX_IACFPropertyMap.h"
#include "AAX.h"
#include "acfunknown.h"
#include "ACFPtr.h"
#include <map>
```

15.314.1 Description

Version-managed concrete PropertyMap class.

Classes

- class [AAX_VPropertyMap](#)
Version-managed concrete [AAX_IPropertyMap](#) class.

15.315 AAX_VPropertyMap.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #ifndef AAX_VPROPERTYMAP_H
00022 #define AAX_VPROPERTYMAP_H
00023
00024 // AAX Includes
00025 #include "AAX_IPropertyMap.h"
00026 #include "AAX_IACFPropertyMap.h"
00027 #include "AAX.h"
00028
00029 // ACF Includes
00030 #include "acfunknown.h"
00031 #include "ACFPtr.h"
00032
00033 // Standard Includes
00034 #include <map>
00035
00036
00037 class IACFComponentFactory;
00038 class AAX_IACFPropertyMap;
00039 class AAX_IACFDescriptionHost;
00040
00045 class AAX_VPropertyMap : public AAX_IPropertyMap
00046 {
00047 public:
00048     // Using static creation methods instead of public constructor in order to
00049     // distinguish between creating a new property map from a component factory
00050     // and acquiring a reference to an existing property map.
00051     static AAX_VPropertyMap* Create ( IACFUnknown* inComponentFactory );
00052     static AAX_VPropertyMap* Acquire ( IACFUnknown* inPropertyMapUnknown );
00053
00054 private:
```

```

00055     AAX_VPropertyMap ();
00056     void InitWithFactory (IACFComponentFactory* inComponentFactory, IACFUnknown* inAuxiliaryUnknown);
00057     void InitWithPropertyMap (IACFUnknown* inPropertyMapUnknown, IACFUnknown* inAuxiliaryUnknown);
00058
00059 public:
00060     ~AAX_VPropertyMap(void) AAX_OVERRIDE;
00061
00062     // AAX_IACFPropertyMap methods
00063     AAX_CBoolean      GetProperty ( AAX_EProperty inProperty, AAX_CPropertyValue * outValue ) const
AAX_OVERRIDE;
00064     AAX_CBoolean      GetPointerProperty ( AAX_EProperty inProperty, const void** outValue ) const
AAX_OVERRIDE;
00065     AAX_Result        AddProperty ( AAX_EProperty inProperty, AAX_CPropertyValue inValue )
AAX_OVERRIDE;
00066     AAX_Result        AddPointerProperty ( AAX_EProperty inProperty, const void* inValue )
AAX_OVERRIDE;
00067     AAX_Result        AddPointerProperty ( AAX_EProperty inProperty, const char* inValue )
AAX_OVERRIDE;
00068     AAX_Result        RemoveProperty ( AAX_EProperty inProperty ) AAX_OVERRIDE;
00069     AAX_Result        AddPropertyWithIDArray ( AAX_EProperty inProperty, const
AAX_SPlugInIdentifierTriad* inPluginIDs, uint32_t inNumPluginIDs) AAX_OVERRIDE;
00070     AAX_CBoolean      GetPropertyWithIDArray ( AAX_EProperty inProperty, const
AAX_SPlugInIdentifierTriad** outPluginIDs, uint32_t* outNumPluginIDs) const AAX_OVERRIDE;
00071
00072     // AAX_IPropertyMap methods
00073     IACFUnknown*      GetIUnknown() AAX_OVERRIDE;
00074
00075 private:
00076     ACFPtr<AAX_IACFPropertyMap> mIACFPropertyMap;
00077     ACFPtr<AAX_IACFPropertyMap_V2> mIACFPropertyMapV2;
00078     ACFPtr<AAX_IACFPropertyMap_V3> mIACFPropertyMapV3;
00079     ACFPtr<AAX_IACFDescriptionHost> mIACFDescriptionHost;
00080     std::map<AAX_EProperty, const void*> mLocalPointerPropertyCache;
00081 };
00082
00083
00084
00085 #endif // AAX_VPROPERTYMAP_H

```

15.316 AAX_VSessionDocument.h File Reference

```

#include "AAX_ISessionDocument.h"
#include "ACFPtr.h"

```

Classes

- class [AAX_VSessionDocument](#)
- class [AAX_VSessionDocument::VTempoMap](#)

Macros

- #define [AAX_VSessionDocument_H](#)

15.316.1 Macro Definition Documentation

15.316.1.1 AAX_VSessionDocument_H

```
#define AAX_VSessionDocument_H
```


15.317 AAX_VSessionDocument.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019 #ifndef AAX_VSessionDocument_H
00020 #define AAX_VSessionDocument_H
00021
00022 #include "AAX_ISessionDocument.h"
00023 #include "ACFPtr.h"
00024
00025 class AAX_IACFSessionDocument;
00026 class AAX_IDataBufferWrapper;
00027
00028 class AAX_VSessionDocument : public AAX_ISessionDocument
00029 {
00030 public:
00031     explicit AAX_VSessionDocument(IACFUnknown * iUnknown);
00032     ~AAX_VSessionDocument() AAX_OVERRIDE;
00033
00034     class VTempoMap : public AAX_ISessionDocument::TempoMap
00035     {
00036     public:
00037         ~VTempoMap() AAX_OVERRIDE;
00038         explicit VTempoMap(IACFUnknown & inDataBuffer);
00039         int32_t Size() const AAX_OVERRIDE;
00040         AAX_CTempoBreakpoint const * Data() const AAX_OVERRIDE;
00041     private:
00042         std::unique_ptr<AAX_IDataBufferWrapper const> mDataBuffer;
00043     };
00044
00048     void Clear();
00049
00050     bool Valid() const AAX_OVERRIDE;
00051     std::unique_ptr<AAX_ISessionDocument::TempoMap const> GetTempoMap() AAX_OVERRIDE;
00052     AAX_Result GetDocumentData(AAX_DocumentData_UID const & inDataType, IACFUnknown ** outData)
00053     AAX_OVERRIDE;
00054 private:
00055     ACFPtr<AAX_IACFSessionDocument> mSessionDocumentV1;
00056 };
00057
00058 #endif // AAX_VSessionDocument_H
```

15.318 AAX_VTask.h File Reference

```
#include "AAX_ITask.h"
#include "AAX.h"
#include "ACFPtr.h"
```

Classes

- class [AAX_VTask](#)
Version-managed concrete [AAX_ITask](#).

Macros

- #define [AAX_VTask_H](#)

15.318.1 Macro Definition Documentation

15.318.1.1 AAX_VTask_H

```
#define AAX_VTask_H
```

15.319 AAX_VTask.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00016 /*=====*/
00017
00018 #pragma once
00019
00020 #ifndef AAX_VTask_H
00021 #define AAX_VTask_H
00022
00023 #include "AAX_ITask.h"
00024 #include "AAX.h"
00025
00026 #include "ACFPtr.h"
00027
00028 class IACFUnknown;
00029
00033 class AAX_VTask : public AAX_ITask
00034 {
00035 public:
00036     explicit AAX_VTask( IACFUnknown* pUnknown );
00037     ~AAX_VTask() AAX_OVERRIDE;
00038
00039     AAX_Result GetType(AAX_CTypeID * oType) const AAX_OVERRIDE;
00040     AAX_IACFDataBuffer const * GetArgumentOfType(AAX_CTypeID iType) const AAX_OVERRIDE;
00041
00042     AAX_Result SetProgress(float iProgress) AAX_OVERRIDE;
00043     float GetProgress() const AAX_OVERRIDE;
00044     AAX_Result AddResult(AAX_IACFDataBuffer const * iResult) AAX_OVERRIDE;
00045     AAX_ITask * SetDone(AAX_TaskCompletionStatus iStatus) AAX_OVERRIDE;
00046 private:
00047     ACFPtr<AAX_IACFTask> mTaskV1;
00048 };
00049
00050 #endif
```

15.320 AAX_VTransport.h File Reference

```
#include "AAX_ITransport.h"
#include "AAX_IACFTransport.h"
#include "AAX_IACFTransportControl.h"
#include "ACFPtr.h"
```

15.320.1 Description

Version-managed concrete Transport class.

Classes

- class [AAX_VTransport](#)

Version-managed concrete [AAX_ITransport](#) class.

15.321 AAX_VTransport.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019-2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #ifndef AAX_VTRANSPORT_H
00022 #define AAX_VTRANSPORT_H
00023
00024 #pragma once
00025
00026 #include "AAX_ITransport.h"
00027 #include "AAX_IACFTransport.h"
00028 #include "AAX_IACFTransportControl.h"
00029 #include "ACFPtr.h"
00030
00035 class AAX_VTransport : public AAX_ITransport
00036 {
00037 public:
00038     AAX_VTransport( IACFUnknown* pUnknown );
00039     ~AAX_VTransport() AAX_OVERRIDE;
00040
00041     // Transport Information Getters
00042     // AAX_IACFTransport
00043     AAX_Result    GetCurrentTempo ( double* TempoBPM ) const AAX_OVERRIDE;
00044     AAX_Result    GetCurrentMeter ( int32_t* MeterNumerator, int32_t* MeterDenominator ) const
00045     AAX_OVERRIDE;
00046     AAX_Result    IsTransportPlaying ( bool* isPlaying ) const AAX_OVERRIDE;
00047     AAX_Result    GetCurrentTickPosition ( int64_t* TickPosition ) const AAX_OVERRIDE;
00048     AAX_Result    GetCurrentLoopPosition ( bool* bLooping, int64_t* LoopStartTick, int64_t*
00049     LoopEndTick ) const AAX_OVERRIDE;
00050     AAX_Result    GetCurrentNativeSampleLocation ( int64_t* SampleLocation ) const AAX_OVERRIDE;
00051     AAX_Result    GetCustomTickPosition( int64_t* oTickPosition, int64_t iSampleLocation) const
00052     AAX_OVERRIDE;
00053     AAX_Result    GetBarBeatPosition(int32_t* Bars, int32_t* Beats, int64_t* DisplayTicks, int64_t
00054     SampleLocation) const AAX_OVERRIDE;
00055     AAX_Result    GetTicksPerQuarter ( uint32_t* ticks ) const AAX_OVERRIDE;
00056     AAX_Result    GetCurrentTicksPerBeat ( uint32_t* ticks ) const AAX_OVERRIDE;
00057     // AAX_IACFTransport_V2
00058     AAX_Result    GetTimelineSelectionStartPosition ( int64_t* oSampleLocation ) const AAX_OVERRIDE;
00059     AAX_Result    GetTimeCodeInfo( AAX_EFrameRate* oFrameRate, int32_t* oOffset ) const AAX_OVERRIDE;
00060     AAX_Result    GetFeetFramesInfo( AAX_EFeetFramesRate* oFeetFramesRate, int64_t* oOffset ) const
00061     AAX_OVERRIDE;
00062     AAX_Result    IsMetronomeEnabled ( int32_t* isEnabled ) const AAX_OVERRIDE;
00063     // AAX_IACFTransport_V3
00064     AAX_Result    GetHDTimeCodeInfo( AAX_EFrameRate* oHDFrameRate, int64_t* oHDOffset ) const
00065     AAX_OVERRIDE;
00066     // AAX_IACFTransport_V4
00067     AAX_Result    GetTimelineSelectionEndPosition( int64_t* oSampleLocation ) const AAX_OVERRIDE;
00068     // AAX_IACFTransportControl

```

```

00067     AAX_Result RequestTransportStart() AAX_OVERRIDE;
00068     AAX_Result RequestTransportStop() AAX_OVERRIDE;
00069
00070 private:
00071     ACFPtr<AAX_IACFTransport>      mITransport;
00072     ACFPtr<AAX_IACFTransport_V2>   mITransportV2;
00073     ACFPtr<AAX_IACFTransport_V3>   mITransportV3;
00074     ACFPtr<AAX_IACFTransport_V4>   mITransportV4;
00075     ACFPtr<AAX_IACFTransportControl> mITransportControl;
00076 };
00077
00078 #endif // AAX_VTRANSPORT_H
00079

```

15.322 AAX_VViewContainer.h File Reference

```

#include "AAX_IViewContainer.h"
#include "AAX_IACFViewContainer.h"
#include "ACFPtr.h"

```

15.322.1 Description

Version-managed concrete ViewContainer class.

Classes

- class [AAX_VViewContainer](#)
Version-managed concrete [AAX_IViewContainer](#) class.

15.323 AAX_VViewContainer.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  *
00004  * Copyright 2013-2017, 2019, 2021, 2023 Avid Technology, Inc.
00005  * All rights reserved.
00006  *
00007  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00008  * not disclose to any third party. Use of the information contained in this
00009  * document is subject to an Avid SDK license.
00010  *
00011  */
00012
00019 /*=====*/
00020
00021 #ifndef AAX_VVIEWCONTAINER_H
00022 #define AAX_VVIEWCONTAINER_H
00023
00024 #include "AAX_IViewContainer.h"
00025 #include "AAX_IACFViewContainer.h"
00026 #include "ACFPtr.h"
00027
00028
00029 class IACFUnknown;
00030
00035 class AAX_VViewContainer : public AAX_IViewContainer
00036 {
00037 public:
00038     AAX_VViewContainer( IACFUnknown * pUnknown );
00039     ~AAX_VViewContainer() AAX_OVERRIDE;
00040
00041     // AAX_IACFViewContainer

```

```

00042
00043 // Getters
00044 int32_t      GetType () AAX_OVERRIDE;
00045 void *       GetPtr () AAX_OVERRIDE;
00046 AAX_Result   GetModifiers ( uint32_t * outModifiers ) AAX_OVERRIDE;
00047
00048 // Setters
00049 AAX_Result   SetViewSize ( AAX_Point & inSize ) AAX_OVERRIDE;
00050 AAX_Result   HandleParameterMouseDown ( AAX_CParamID inParamID, uint32_t inModifiers )
AAX_OVERRIDE;
00051 AAX_Result   HandleParameterMouseDrag ( AAX_CParamID inParamID, uint32_t inModifiers )
AAX_OVERRIDE;
00052 AAX_Result   HandleParameterMouseUp ( AAX_CParamID inParamID, uint32_t inModifiers )
AAX_OVERRIDE;
00053 AAX_Result   HandleParameterMouseEnter ( AAX_CParamID inParamID, uint32_t inModifiers )
AAX_OVERRIDE;
00054 AAX_Result   HandleParameterMouseExit ( AAX_CParamID inParamID, uint32_t inModifiers )
AAX_OVERRIDE;
00055 AAX_Result   HandleMultipleParametersMouseDown ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) AAX_OVERRIDE;
00056 AAX_Result   HandleMultipleParametersMouseDrag ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) AAX_OVERRIDE;
00057 AAX_Result   HandleMultipleParametersMouseUp ( const AAX_CParamID* inParamIDs, uint32_t
inNumOfParams, uint32_t inModifiers ) AAX_OVERRIDE;
00058
00059 private:
00060 ACFPtr<AAX_IACFViewContainer>      mIViewContainer;
00061 ACFPtr<AAX_IACFViewContainer_V2>    mIViewContainerV2;
00062 ACFPtr<AAX_IACFViewContainer_V3>    mIViewContainerV3;
00063 };
00064
00065
00066 #endif //AAX_VVIEWCONTAINER_H

```

15.324 AAX_Alignment.h File Reference

```
#include <stddef.h>
```

15.324.1 Description

Alignment malloc and free methods for optimization.

Namespaces

- namespace [AAX](#)

Functions

- void [AAX::alignFree](#) (void *p)
- template<class T >
T * [AAX::alignMalloc](#) (int iArraySize, int iAlignment)

15.325 AAX_Alignment.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   * Copyright 2009-2015, 2023 Avid Technology, Inc.
00004   * All rights reserved.
00005   *
00006   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007   * not disclose to any third party. Use of the information contained in this
00008   * document is subject to an Avid SDK license.
00009   */
00010
00017  /*=====*/
00018
00019  #ifndef AAX_ALIGNMENT_H
00020  #define AAX_ALIGNMENT_H
00021
00022  #include <stddef.h>
00023
00024  namespace AAX
00025  {
00026
00027      inline void alignFree(void *p)
00028      {
00029          char** aTempPtr=reinterpret_cast<char**>(p);
00030          aTempPtr--; //backup 4 bytes past the beginning of the buffer
00031          char* aRealPtr = aTempPtr[0]; //Get the real address
00032
00033          if(aRealPtr)
00034              ::delete[] aRealPtr;
00035      }
00036
00037      template <class T>
00038      T* alignMalloc(int iArraySize, int iAlignment)
00039      {
00040          // We can seriously mess ourselves up if alignment is not a power of 2
00041          if ((iAlignment != 2) && (iAlignment != 4) && (iAlignment != 8) && (iAlignment != 16) &&
00042              (iAlignment != 32)) {
00043              return 0;
00044          }
00045          // We can't allocate a negative-size array
00046          if (iArraySize <= 0) {
00047              return 0;
00048          }
00049          const unsigned int cSizeOfPointer = sizeof(char*);
00050          // Over-allocate memory by the maximum offset we could be from our requested alignment
00051          char* aRealPtr = ::new char[iArraySize*sizeof(T) + iAlignment + cSizeOfPointer];
00052          if (!aRealPtr) {
00053              return 0;
00054          }
00055          char* p=aRealPtr;
00056          p+=cSizeOfPointer; //Skip four bytes (we store the real base address here)
00057          size_t mod = size_t(p) & (iAlignment-1);
00058          if (mod)
00059              p += (iAlignment - mod);
00060          *reinterpret_cast<char**>(p-cSizeOfPointer)=aRealPtr; //Save the real address. We'll need
00061          it for delete.
00062          return (T*) p;
00063      }
00064  } // namespace AAX
00065  #endif //AAX_ALIGNMENT_H

```

15.326 AAX_Constants.h File Reference

```
#include <cmath>
```

15.326.1 Description

Signal processing constants.

Namespaces

- namespace [AAX](#)

Macros

- `#define` [AAX_CONSTANTS_H](#)

Enumerations

- enum [AAX::ESampleRates](#) {
[AAX::e44100SampleRate](#) = 44100 ,
[AAX::e48000SampleRate](#) = 48000 ,
[AAX::e88200SampleRate](#) = 88200 ,
[AAX::e96000SampleRate](#) = 96000 ,
[AAX::e176400SampleRate](#) = 176400 ,
[AAX::e192000SampleRate](#) = 192000 }

Variables

- const int [AAX::cBigEndian](#) =0
- const int [AAX::cLittleEndian](#) =1
- const double [AAX::cPi](#) = 3.1415926535897932384626433832795
- const double [AAX::cTwoPi](#) = 6.2831853071795862319959269370884
- const double [AAX::cHalfPi](#) = 1.5707963267948965579989817342721
- const double [AAX::cQuarterPi](#) = 0.78539816339744827899949086713605
- const double [AAX::cRootTwo](#) = 1.4142135623730950488016887242097
- const double [AAX::cOneOverRootTwo](#) = 0.70710678118654752440084436210485
- const double [AAX::cPos3dB](#) =1.4142135623730950488016887242097
- const double [AAX::cNeg3dB](#) =0.70710678118654752440084436210485
- const double [AAX::cPos6dB](#) =2.0
- const double [AAX::cNeg6dB](#) =0.5
- const double [AAX::cNormalizeLongToAmplitudeOneHalf](#) = 0.00000000023283064365386962890625
- const double [AAX::cNormalizeLongToAmplitudeOne](#) = 1.0/double(1<<31)
- const double [AAX::cMilli](#) =0.001
- const double [AAX::cMicro](#) =0.001*0.001
- const double [AAX::cNano](#) =0.001*0.001*0.001
- const double [AAX::cPico](#) =0.001*0.001*0.001*0.001
- const double [AAX::cKilo](#) =1000.0
- const double [AAX::cMega](#) =1000.0*1000.0
- const double [AAX::cGiga](#) =1000.0*1000.0*1000.0

15.326.2 Macro Definition Documentation

15.326.2.1 AAX_CONSTANTS_H

```
#define AAX_CONSTANTS_H
```

15.327 AAX_Constants.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   * Copyright 2009-2015, 2023 Avid Technology, Inc.
00004   * All rights reserved.
00005   *
00006   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007   * not disclose to any third party. Use of the information contained in this
00008   * document is subject to an Avid SDK license.
00009   */
00010
00017  /*=====*/
00018  #pragma once
00019
00020  #ifndef AAX_CONSTANTS_H
00021  #define AAX_CONSTANTS_H
00022
00023
00024  /* the following lines were re-introduced on 6/11/09 because
00025     the FFmt project still uses SInt32 types */
00026  #ifndef _TMS320C6X
00027     typedef signed int SInt32;
00028  #else
00029     // #include "DigiPublicTypes.h"
00030  #endif
00031  /* end 6/11/09 changes */
00032
00033
00034  // Standard headers
00035  #include <cmath>
00036
00037  namespace AAX
00038  {
00039
00040  #if __BIG_ENDIAN__
00041     const int cBigEndian=1;
00042     const int cLittleEndian=0;
00043  #else
00044     const int cBigEndian=0;
00045     const int cLittleEndian=1;
00046  #endif
00047
00048  const double cPi = 3.1415926535897932384626433832795;
00049  const double cTwoPi = 6.2831853071795862319959269370884;
00050  //2.0*3.1415926535897932384626433832795;
00051  const double cHalfPi = 1.5707963267948965579989817342721;
00052  //0.5*3.1415926535897932384626433832795;
00053  const double cQuarterPi = 0.78539816339744827899949086713605;
00054  //0.25*3.1415926535897932384626433832795;
00055  const double cRootTwo = 1.4142135623730950488016887242097;
00056  const double cOneOverRootTwo= 0.70710678118654752440084436210485;
00057
00058  //Obviously these numbers are are not exact.
00059  const double cPos3dB=1.4142135623730950488016887242097;
00060  const double cNeg3dB=0.70710678118654752440084436210485;
00061  const double cPos6dB=2.0;
00062  const double cNeg6dB=0.5;
00063
00064  const double cNormalizeLongToAmplitudeOneHalf = 0.00000000023283064365386962890625;
00065  //1.0/double(1LL<<32LL);
00066  const double cNormalizeLongToAmplitudeOne = 1.0/double(1<<31);
00067  //-0.0000000004656612873077392578125;
00068
00069  const double cMilli=0.001;
00070  const double cMicro=0.001*0.001;
00071  const double cNano=0.001*0.001*0.001;
00072  const double cPico=0.001*0.001*0.001*0.001;
00073
00074  const double cKilo=1000.0;
00075  const double cMega=1000.0*1000.0;
00076  const double cGiga=1000.0*1000.0*1000.0;
00077
00078  enum ESampleRates
00079  {
00080     e44100SampleRate = 44100,
00081     e48000SampleRate = 48000,
00082     e88200SampleRate = 88200,
00083     e96000SampleRate = 96000,
00084     e176400SampleRate = 176400,
00085     e192000SampleRate = 192000
00086  };
00087
00088

```



```
00084 } // namespace AAX
00085
00086 #endif // AAX_CONSTANTS_H
00087
```

15.328 AAX_Denormal.h File Reference

```
#include "AAX.h"
#include "AAX_PlatformOptimizationConstants.h"
#include <limits>
#include <math.h>
```

15.328.1 Description

Signal processing utilities for denormal/subnormal floating point numbers.

Namespaces

- namespace [AAX](#)

Macros

- #define [AAX_DENORMAL_H](#)
- #define [AAX_SCOPE_COMPUTE_DENORMALS\(\)](#) do {} while(0)
Sets the run-time environment to handle denormal floats within the scope of this macro.
- #define [AAX_SCOPE_DENORMALS_AS_ZERO\(\)](#) do {} while(0)
Sets the run-time environment to treat denormal floats as zero within the scope of this macro.

Functions

- void [AAX::DeDenormal](#) (double &iValue)
Clamps very small floating point values to zero.
- void [AAX::DeDenormal](#) (float &iValue)
Clamps very small floating point values to zero.
- void [AAX::DeDenormalFine](#) (float &iValue)
- void [AAX::FilterDenormals](#) (float *inSamples, int32_t inLength)
Round all denormal/subnormal samples in a buffer to zero.

Variables

- const double [AAX::cDenormalAvoidanceOffset](#) =3.0e-34
- const float [AAX::cFloatDenormalAvoidanceOffset](#) =3.0e-20f

15.328.2 Macro Definition Documentation

15.328.2.1 AAX_DENORMAL_H

```
#define AAX_DENORMAL_H
```

15.328.2.2 AAX_SCOPE_COMPUTE_DENORMALS

```
#define AAX_SCOPE_COMPUTE_DENORMALS( ) do {} while(0)
```

Sets the run-time environment to handle denormal floats within the scope of this macro.

The host sets the denormal policy for all [AAX](#) threads and may use settings which treat denormal float values as zero (DAZ+FZ). This macro forces non-DAZ behavior.

15.328.2.3 AAX_SCOPE_DENORMALS_AS_ZERO

```
#define AAX_SCOPE_DENORMALS_AS_ZERO( ) do {} while(0)
```

Sets the run-time environment to treat denormal floats as zero within the scope of this macro.

The host sets the denormal policy for all [AAX](#) threads and may already use settings which treat denormal float values as zero (DAZ+FZ). This macro forces DAZ behavior.

15.329 AAX_Denormal.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2014-2016, 2019, 2021, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00018 #pragma once
00019
00020 #ifndef AAX_DENORMAL_H
00021 #define AAX_DENORMAL_H
00022
00023
00024 // AAX Includes
00025 #include "AAX.h"
00026 #include "AAX_PlatformOptimizationConstants.h"
00027
00028
00029 #if ! defined( _TMS320C6X )
00030 #include <limits>
00031 #include <math.h>
00032 #endif // ! defined( _TMS320C6X )
00033
00034
00035
00053 #if defined( _TMS320C6X )
00054
00055 // TODO: Currently noop on TI
00056 #define AAX_SCOPE_COMPUTE_DENORMALS() do {} while(0)
00057 #define AAX_SCOPE_DENORMALS_AS_ZERO() do {} while(0)
00058
00059 #elif ((defined _MSC_VER) || defined(WINDOWS_VERSION))
00060 // Visual Studio
```

```

00061
00062 // Note: In Pro Tools 11 and higher for Windows, DAZ is turned on for all plug-in processing threads
00063
00064 // FIXME: These macros are currently noop on Windows because of compiler problems when intrin.h is
00065 // included. Investigate using _controlfp(_DN_FLUSH, _MCW_DN) instead
00066 #define AAX_SCOPE_COMPUTE_DENORMALS() do {} while(0)
00067 #define AAX_SCOPE_DENORMALS_AS_ZERO() do {} while(0)
00068 // #define AAX_SCOPE_COMPUTE_DENORMALS() AAX_StDenormalAsZeroFlagsOff computeDenormalFlagsScope ##
00069 // _LINE_ ; do {} while (0)
00070 // #define AAX_SCOPE_DENORMALS_AS_ZERO() AAX_StDenormalAsZeroFlagsOn denormalAsZeroFlagsScope ##
00071 // _LINE_ ; do {} while (0)
00072 //
00073 // #include <Windows.h>
00074 // #include <mmintrin.h>
00075 //
00076 // const unsigned gMXCSRFlags_DAZFZ = 0x00008040;
00077 //
00078 // struct AAX_StDenormalAsZeroFlagsOn
00079 // {
00080 // public:
00081 //     AAX_StDenormalAsZeroFlagsOn()
00082 //     : mSSESupported(IsProcessorFeaturePresent(PF_XMMI_INSTRUCTIONS_AVAILABLE))
00083 //     {
00084 //         if (mSSESupported) { set_flags(); }
00085 //     }
00086 //     ~AAX_StDenormalAsZeroFlagsOn()
00087 //     {
00088 //         if (mSSESupported) { reset_flags(); }
00089 //     }
00090 // private:
00091 //     void set_flags()
00092 //     {
00093 //         mPrevFlags = _mm_getcsr();
00094 //         unsigned newFlags = mPrevFlags | gMXCSRFlags_DAZFZ;
00095 //         _mm_setcsr(newFlags);
00096 //     }
00097 //     void reset_flags()
00098 //     {
00099 //         _mm_setcsr(mPrevFlags);
00100 //     }
00101 //     unsigned mPrevFlags;
00102 //     const bool mSSESupported;
00103 // };
00104 //
00105 // struct AAX_StDenormalAsZeroFlagsOff
00106 // {
00107 // public:
00108 //     AAX_StDenormalAsZeroFlagsOff()
00109 //     : mSSESupported(IsProcessorFeaturePresent(PF_XMMI_INSTRUCTIONS_AVAILABLE))
00110 //     {
00111 //         if (mSSESupported) { set_flags(); }
00112 //     }
00113 //     ~AAX_StDenormalAsZeroFlagsOff()
00114 //     {
00115 //         if (mSSESupported) { reset_flags(); }
00116 //     }
00117 // private:
00118 //     void set_flags()
00119 //     {
00120 //         mPrevFlags = _mm_getcsr();
00121 //         unsigned newFlags = mPrevFlags & (~gMXCSRFlags_DAZFZ);
00122 //         _mm_setcsr(newFlags);
00123 //     }
00124 //     void reset_flags()
00125 //     {
00126 //         _mm_setcsr(mPrevFlags);
00127 //     }
00128 //     unsigned mPrevFlags;
00129 //     const bool mSSESupported;
00130 // };
00131 //
00132 #elif defined(__linux__)
00133 // Linux
00134 //
00135 // TODO: Currently noop on Linux
00136 #define AAX_SCOPE_COMPUTE_DENORMALS() do {} while(0)
00137 #define AAX_SCOPE_DENORMALS_AS_ZERO() do {} while(0)
00138

```

```

00152 #elif (defined (__GNUC__) || defined(MAC_VERSION))
00153 // GCC / macOS
00154
00155 // Note: In Pro Tools 11 through Pro Tools 12.6 on macOS, DAZ is turned off for all plug-in processing
        threads
00156
00157 #define AAX_SCOPE_COMPUTE_DENORMALS() AAX_StDenormalAsZeroFlagsOff computeDenormalFlagsScope ##
        __LINE__ ; do {} while (0)
00158 #define AAX_SCOPE_DENORMALS_AS_ZERO() AAX_StDenormalAsZeroFlagsOn denormalAsZeroFlagsScope ## __LINE__
        ; do {} while (0)
00159
00160 #include <fenv.h>
00161 struct AAX_StDenormalAsZeroFlagsOn
00162 {
00163 public:
00164     AAX_StDenormalAsZeroFlagsOn() { set_flags(); }
00165     ~AAX_StDenormalAsZeroFlagsOn() { reset_flags(); }
00166
00167 private:
00168     #if !defined(__arm64__)
00169         void set_flags() { fegetenv(&mPrevFlags); fesetenv(FE_DFL_DISABLE_SSE_DENORMS_ENV); }
00170     #else
00171         void set_flags() { fegetenv(&mPrevFlags); fesetenv(FE_DFL_DISABLE_DENORMS_ENV); }
00172     #endif
00173     void reset_flags() { fesetenv(&mPrevFlags); }
00174     fenv_t mPrevFlags;
00175 };
00176 struct AAX_StDenormalAsZeroFlagsOff
00177 {
00178 public:
00179     AAX_StDenormalAsZeroFlagsOff() { set_flags(); }
00180     ~AAX_StDenormalAsZeroFlagsOff() { reset_flags(); }
00181
00182 private:
00183     void set_flags() { fegetenv(&mPrevFlags); fesetenv(FE_DFL_ENV); } // assuming that FE_DFL_ENV are
        the flags that we want here
00184     void reset_flags() { fesetenv(&mPrevFlags); }
00185     fenv_t mPrevFlags;
00186 };
00187
00188
00189
00190 #else
00191
00192 #error AAX_SCOPE_COMPUTE_DENORMALS is not defined for this compiler
00193 #error AAX_SCOPE_DENORMALS_AS_ZERO is not defined for this compiler
00194
00195 // Just for Doxygen
00196 #define AAX_SCOPE_COMPUTE_DENORMALS() do {} while(0)
00197 #define AAX_SCOPE_DENORMALS_AS_ZERO() do {} while(0)
00198
00199 #endif // End compiler selection for AAX_SCOPE_COMPUTE_DENORMALS / AAX_SCOPE_DENORMALS_AS_ZERO
        definition
00200
00201
00202
00203
00204 namespace AAX
00205 {
00206     const double cDenormalAvoidanceOffset=3.0e-34;
00207     const float cFloatDenormalAvoidanceOffset=3.0e-20f; //This number is empirically derived with
        minimal trial and error.
00208
00209 #if ! defined( _TMS320C6X )
00210     static const float cMinimumNormalFloat = std::numeric_limits<float>::min();
00211 #endif
00212
00222     inline void DeDenormal(double &iValue)
00223     {
00224 #if defined(WINDOWS_VERSION) || defined(MAC_VERSION) || defined(LINUX_VERSION)
00225         if(iValue < cDenormalAvoidanceOffset && iValue > -cDenormalAvoidanceOffset) iValue=0.0;
00226 #endif
00227     }
00228
00231     inline void DeDenormal(float &iValue)
00232     {
00233 #if defined(WINDOWS_VERSION) || defined(MAC_VERSION) || defined(LINUX_VERSION)
00234         if(iValue < cFloatDenormalAvoidanceOffset && iValue > -cFloatDenormalAvoidanceOffset)
            iValue=0.0f;
00235 #endif
00236     }
00237
00238 #if AAX_CPP11_SUPPORT
00241     inline float DeDenormal(float &&iValue)
00242     {
00243 #if defined(WINDOWS_VERSION) || defined(MAC_VERSION) || defined(LINUX_VERSION)
00244         return (iValue < cFloatDenormalAvoidanceOffset && iValue > -cFloatDenormalAvoidanceOffset) ?

```

```

        0.f : iValue;
00245 #endif
00246     }
00247 #endif // AAX_CPP11_SUPPORT
00248
00253     inline void DeDenormalFine(float &iValue)
00254     {
00255     #if ! defined( _TMS320C6X )
00256         if(iValue < cMinimumNormalFloat && iValue > -cMinimumNormalFloat) iValue=0.0f;
00257     #endif
00258     }
00259
00270     inline void FilterDenormals(float* inSamples, int32_t inLength)
00271     {
00272         //TODO: Implement optimized versions for TI and SSE
00273     #if ! defined( _TMS320C6X ) // Not yet implemented for TI
00274         for( int32_t i = 0; i < inLength; ++i )
00275         {
00276             float curFloat = *inSamples;
00277             if( fabsf(curFloat) < cMinimumNormalFloat )
00278                 curFloat = 0.0f;
00279             *(inSamples++) = curFloat;
00280         }
00281     #endif
00282     }
00283
00284 } // namespace AAX
00285
00286 #endif // AAX_QUANTIZE_H

```

15.330 AAX_FastInterpolatedTableLookup.h File Reference

```

#include "AAX_Quantize.h"
#include <AAX_ALIGN_FILE_BEGIN>
#include <AAX_ALIGN_FILE_ALG>
#include <AAX_ALIGN_FILE_END>
#include <AAX_ALIGN_FILE_RESET>
#include "AAX_FastInterpolatedTableLookup.hpp"

```

15.330.1 Description

A set of functions that provide lookup table functionality. Not necessarily optimized for TI, but used internally.

Classes

- class [AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >](#)

Macros

- #define [AAX_FASTINTERPOLATEDTABLELOOKUP_H](#)

15.330.2 Macro Definition Documentation

15.330.2.1 AAX_FASTINTERPOLATEDTABLELOOKUP_H

```
#define AAX_FASTINTERPOLATEDTABLELOOKUP_H
```

15.331 AAX_FastInterpolatedTableLookup.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2013-2015, 2019, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010 /*=====*/
00011 #pragma once
00012
00013 #ifndef AAX_FASTINTERPOLATEDTABLELOOKUP_H
00014 #define AAX_FASTINTERPOLATEDTABLELOOKUP_H
00015
00016 #include "AAX_Quantize.h"
00017
00018 //=====
00019 // NOTE:
00020 // Constructors and destructors are not call because state and coefficients blocks
00021 // are not allocated as classes
00022 //
00023 #include AAX_ALIGN_FILE_BEGIN
00024 #include AAX_ALIGN_FILE_ALG
00025 #include AAX_ALIGN_FILE_END
00026
00027 template<class TFLOAT, class DFLOAT>
00028 class AAX_FastInterpolatedTableLookup
00029 {
00030 public:
00031     void SetParameters(int iTableSize, TFLOAT iMin=0.0, TFLOAT iMax=1.0, int iNumTables=1)
00032     {
00033         mTableSizeM1=(TFLOAT) (iTableSize-1);
00034         mTableSize=(TFLOAT) iTableSize;
00035         mIntTableSizeM1=iTableSize-1;
00036         mMin=iMin;
00037         mMax=iMax;
00038         mMaxMinusMin=iMax-iMin;
00039         mTableSizeM1DivMaxMinusMin=TFLOAT(iTableSize-1)/(iMax-iMin); //Two divides??
00040         mTableSizeDivMaxMinusMin=TFLOAT(iTableSize)/(iMax-iMin);
00041         mNumTables=iNumTables;
00042     }
00043
00044     DFLOAT DoTableLookupExtraFast(const TFLOAT* const iTable, DFLOAT iValue) const;
00045     void DoTableLookupExtraFastMulti(const TFLOAT* iTable, DFLOAT iValue, DFLOAT* oValues) const;
00046
00047     void DoTableLookupExtraFast(const TFLOAT* const iTable, const TFLOAT* const inpBuf, DFLOAT*
00048     const outBuf, int blockSize);
00049
00050     TFLOAT GetMin() { return mMin; };
00051     TFLOAT GetMaxMinusMin() { return mMaxMinusMin; };
00052
00053 private:
00054     TFLOAT mTableSizeM1;
00055     int mIntTableSizeM1;
00056     TFLOAT mTableSizeM1DivMaxMinusMin;
00057     TFLOAT mMin;
00058     TFLOAT mMax;
00059     TFLOAT mMaxMinusMin;
00060     TFLOAT mTableSize;
00061     TFLOAT mTableSizeDivMaxMinusMin;
00062     int mNumTables; //This is used for multi-element lookups
00063 };
00064
00065 #include AAX_ALIGN_FILE_BEGIN
00066 #include AAX_ALIGN_FILE_RESET
00067 #include AAX_ALIGN_FILE_END
```

```

00106
00107 // Template implementation
00108 #include "AAX_FastInterpolatedTableLookup.hpp"
00109
00110 #endif // AAX_FASTINTERPOLATEDTABLELOOKUP_H

```

15.332 AAX_FastInterpolatedTableLookup.hpp File Reference

```
#include "AAX_Quantize.h"
```

15.333 AAX_FastInterpolatedTableLookup.hpp

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2009-2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010 /*=====*/
00011
00012 #ifndef AAX_FASTINTERPOLATEDTABLELOOKUP_HPP
00013 #define AAX_FASTINTERPOLATEDTABLELOOKUP_HPP
00014
00015 #include "AAX_Quantize.h"
00016
00017 //This version requires that the lookup table is padded with one extra location so we
00018 //can avoid one of the checks to see if ours pointers are out of bounds.
00019 template<class TFLOAT, class DFLOAT>
00020 inline DFLOAT AAX_FastInterpolatedTableLookup<TFLOAT,DFLOAT>::DoTableLookupExtraFast(const TFLOAT*
    iTable, DFLOAT iValue) const
00021 {
00022     const TFLOAT aScaledValue=(iValue - mMin)*mTableSizeM1DivMaxMinusMin;
00023
00024     //Clamp between 0.0 and table size, so we don't go out of the table bounds. One thing that's not
    obvious is this clamp was written
00025     //so that it will return 0.0 if a NaN is given as the table input. This keeps us within bounds
    even if we're feed garbage.;
00026     const TFLOAT aScaledValueLimited = aScaledValue > mTableSizeM1 ? mTableSizeM1 : (aScaledValue >
    0.0f ? aScaledValue : 0.0f);
00027
00028     int aTableIndex=AAX::FastTrunc2Int32(aScaledValueLimited);
00029
00030     TFLOAT aFracIndex=aScaledValueLimited - TFLOAT(aTableIndex);
00031     TFLOAT aFracIndexM1=1.0f-aFracIndex;
00032
00033     return (DFLOAT) (iTable[aTableIndex]*aFracIndexM1 + iTable[aTableIndex+1]*aFracIndex);
00034 }
00035
00036 //This version requires that the lookup table is padded with one extra location so we
00037 //can avoid one of the checks to see if ours pointers are out of bounds.
00038 template<class TFLOAT, class DFLOAT>
00039 inline void AAX_FastInterpolatedTableLookup<TFLOAT,DFLOAT>::DoTableLookupExtraFastMulti(const TFLOAT*
    iTable, DFLOAT iValue, DFLOAT* oValues) const
00040 {
00041     TFLOAT aScaledValue=(iValue - mMin)*mTableSizeM1DivMaxMinusMin;
00042
00043     //Clamp between 0.0 and table size, so we don't go out of the table bounds. One thing that's not
    obvious is this clamp was written
00044     //so that it will return 0.0 if a NaN is given as the table input. This keeps us within bounds
    even if we're feed garbage.
00045     const TFLOAT aScaledValueLimited = aScaledValue > mTableSizeM1 ? mTableSizeM1 : (aScaledValue >
    0.0f ? aScaledValue : 0.0f);
00046
00047     int aTableIndex=AAX::FastTrunc2Int32(aScaledValueLimited);
00048
00049     TFLOAT aFracIndex=aScaledValueLimited - TFLOAT(aTableIndex);
00050     TFLOAT aFracIndexM1=1.0f-aFracIndex;
00051
00052     aTableIndex=aTableIndex*mNumTables;
00053     int aTableIndexPlus1=aTableIndex+mNumTables;

```

```

00054
00055     for(int i=0; i<mNumTables; i++)
00056     {
00057         oValues[i]=(DFLOAT) (iTable[aTableIndex++] * aFracIndexM1 +
00058                               iTable[aTableIndexPlus1++] * aFracIndex);
00059     }
00060 }
00061 //Block version of table lookup
00062 template<class TFLOAT, class DFLOAT>
00063 inline void AAX_FastInterpolatedTableLookup<TFLOAT,DFLOAT>::DoTableLookupExtraFast(const TFLOAT* const
00064 iTable, const TFLOAT* const inpBuf, DFLOAT* const outBuf, int blockSize)
00065 {
00066     #if defined( _TMS320C6700 )
00067         const int r = 16;
00068         #pragma MUST_ITERATE(r, , r);
00069     #endif
00070     for (int i = 0; i < blockSize; i++)
00071     {
00072         outBuf[i] = DoTableLookupExtraFast(iTable, inpBuf[i]);
00073     }
00074     return;
00075 }
00076 #endif // AAX_FASTINTERPOLATEDTABLELOOKUP_HPP

```

15.334 AAX_FastPow.h File Reference

```

#include <cmath>
#include "AAX.h"

```

15.334.1 Description

Set of functions to optimize pow.

To use:

```

const int kPowTableExtent = 9;           // Lower values are less precise. 9 is the maximum
float powTableH[kPowTableSize];
float powTableL[kPowTableSize];
int radix = 2;                           // This should be whatever radix you want. Ie: radix ^ exp
PowFastSetTable( powTableH, kPowExtent, kPowExtent, false ); // Set the high table
PowFastSetTable( powTableL, kPowExtent*2, kPowExtent, true ); // Set the low table
..
result = powFastLookup(exp, log(2) / log(radix), powTableH, powTableL);

```

Namespaces

- namespace [AAX](#)

Macros

- #define [_AAX_FASTPOW_H_](#)

Variables

- const unsigned int [AAX::kPowExtent](#) = 9
- const unsigned int [AAX::kPowTableSize](#) = 1 << kPowExtent

15.334.2 Macro Definition Documentation

15.334.2.1 _AAX_FASTPOW_H_

```
#define _AAX_FASTPOW_H_
```

15.335 AAX_FastPow.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2009-2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00029 /*=====*/
00030
00031
00032
00033
00034 #pragma once
00035
00036 #ifndef _AAX_FASTPOW_H_
00037 #define _AAX_FASTPOW_H_
00038
00039 #include <cmath>
00040 #if defined(MAC_VERSION) || defined(LINUX_VERSION)
00041 namespace std {using ::powf;using ::log10f;using ::fabsf;}
00042 #endif
00043 #include "AAX.h" // For AAX_RESTRICT
00044
00045 namespace AAX
00046 {
00047     static const float _2p23 = 8388608.0f;
00048     const unsigned int kPowExtent = 9; // 9 results in ~-80dB cancellation
00049     const unsigned int kPowTableSize = 1 « kPowExtent;
00050
00065     static inline void PowFastSetTable(
00066         float* const AAX_RESTRICT pTable,
00067         const unsigned int precision,
00068         const unsigned int extent,
00069         const bool isRound)
00070     {
00071         // step along table elements and x-axis positions
00072         float zeroToOne = !isRound ?
00073             0.0f : (1.0f / (static_cast<float>(1 « precision) * 2.0f));
00074         for( int i = 0; i < (1 « extent); ++i )
00075         {
00076             // make y-axis value for table element
00077             pTable[i] = std::powf( 2.0f, zeroToOne );
00078             zeroToOne += 1.0f / static_cast<float>(1 « precision);
00080         }
00081     }
00082
00128     static inline float PowFastLookup(
00129         const float val,
00130         const float ilog2,
00131         const float* const AAX_RESTRICT tableH,
00132         const float* const AAX_RESTRICT tableL)
00133     {
00134         // build float bits
00135         const int i = static_cast<int>(( val * (_2p23 * ilog2)) + (127.0f * _2p23) );
00136
00137         // replace mantissa with combined lookups
00138         const float t = tableH[(i « 14) & 0x1FFF] * tableL[(i « 5) & 0x1FFF];
00139         const int it = (i & 0xFF800000) |
00140             (*reinterpret_cast<const int*>(&t) & 0x7FFFFF);
00141     }
```

```

00142
00143 // convert bits to float
00144 return *reinterpret_cast<const float*>( &it );
00145
00146 }
00147
00159 static inline float Pow2FastLookup(
00160     const float      val,
00161     const float* const AAX_RESTRICT tableH,
00162     const float* const AAX_RESTRICT tableL)
00163 {
00164
00165     // build float bits
00166     const int i = static_cast<int>((val + 127.0f) * _2p23);
00167
00168     // replace mantissa with combined lookups
00169     const float t = tableH[(i >> 14) & 0x1FF] * tableL[(i >> 5) & 0x1FF];
00170     const int  it = (i & 0xFF800000) |
00171         (*reinterpret_cast<const int*>( &t ) & 0x007FFFFF);
00172
00173     // convert bits to float
00174     return *reinterpret_cast<const float*>( &it );
00175
00176 }
00177
00178
00179
00180 static const int kMantissaBitSize = 23;
00181 static const int kExpBias = 127;
00182 static const int kLogMantissaMSBs = 9;
00183 static const int kLogTableSize = (1 << kLogMantissaMSBs) + 1; // 513;
00184
00195 static inline void LogFastSetTable(
00196     float* const AAX_RESTRICT logTable,
00197     int tableSize = kLogTableSize)
00198 {
00199     const float kInv = 1.0f/float(tableSize - 1); // 0.001953125
00200     float mantissaVal = 1.0f; //Start here.
00201     const float invLog10Base2 = 1.0f / std::log10f(2.0f);
00202     for (int j = 0; j < tableSize; j++)
00203     {
00204         logTable[j] = std::log10f(mantissaVal) * invLog10Base2; //By log equivalency: log2(x) =
log10(x)/log10(2)
00205         mantissaVal += kInv;
00206     }
00207
00208
00209     return;
00210 }
00211
00230 template <int kMantMSBs>
00231 static inline void LogFloatToExpMantissa(
00232     float number,
00233     int * const AAX_RESTRICT outExp,
00234     int * const AAX_RESTRICT outMant,
00235     float* const AAX_RESTRICT fract)
00236 {
00237     const int mantissaSize = 1 << (kMantissaBitSize - kMantMSBs);
00238     const float invMantissaSize = 1.0f/(float)mantissaSize;
00239     const int intEquiv = *reinterpret_cast<const int*>(&number);
00240
00241     //bool isNegative = (result & 0x80000000) != 0; //Sign bit
00242     const int exponent = (intEquiv & 0x7f800000) >> 23; //Exponent bits
00243     const int mantissa = intEquiv & 0x007fffff; //Mantissa bits.
00244
00245     // 0 000|0 000|0 1 010| 0000 0000 0000 0000 0000
00246     // 0111 1111 1000 0000 0000 0000 0000 0000 = 7f800000 (Exp)
00247
00248     // 0000 0000 0111 1111 1111 1111 1111 1111 = 007fffff (Mant)
00249
00249     *outExp = exponent - kExpBias; //Compensate for exponent bias for caller.
00250     //To do: add halfLSBShifted to mantissa before shifting; *outMant = (mantissa + halfLSBShifted) >>
(kMantissaBitSize - mantMSBs)
00251     *outMant = mantissa >> (kMantissaBitSize - kMantMSBs); //Shift to provide only mantMSBs for
log LUT.
00252     // Provide a linear interpolation fraction
00253     *fract = (float)(mantissa & (mantissaSize - 1)) * invMantissaSize;
00254
00255     return;
00256 }
00257
00271 template <int kPrecision>
00272 static inline float LogFastLookup(
00273     float num,
00274     const float* const AAX_RESTRICT logTable)
00275 {
00276
00277     int exponent, mantissa;

```

```
00278     float fract;
00279
00280     AAX::LogFloatToExpMantissa<kPrecision>(num, &exponent, &mantissa, &fract);
00281
00282     const float mantLog1 = logTable[mantissa];
00283     const float mantLog2 = logTable[mantissa+1];
00284
00285     const float logOfNum = (mantLog1 * (1.0f - fract) + fract * mantLog2) + exponent;
00286     return logOfNum;
00287 }
00288
00289
00290
00291 } // namespace AAX
00292
00293 #endif // #ifndef _AAX_FASTPOW_H_
```

15.336 AAX_Map.h File Reference

```
#include "AAX.h"
#include <AAX_ALIGN_FILE_BEGIN>
#include <AAX_ALIGN_FILE_ALG>
#include <AAX_ALIGN_FILE_END>
#include <AAX_ALIGN_FILE_RESET>
```

15.336.1 Description

Author

Mykola Kryvonos

Classes

- class [AAX_Map](#)

Macros

- #define [AAX_MAP_H](#)

15.336.2 Macro Definition Documentation

15.336.2.1 AAX_MAP_H

```
#define AAX_MAP_H
```

15.337 AAX_Map.h

[Go to the documentation of this file.](#)

```

00001  /*=====*/
00002  /*
00003   * Copyright 2009-2015, 2019, 2023 Avid Technology, Inc.
00004   * All rights reserved.
00005   *
00006   * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007   * not disclose to any third party. Use of the information contained in this
00008   * document is subject to an Avid SDK license.
00009   */
00010
00017  /*=====*/
00018
00019  #pragma once
00020
00021  #ifndef AAX_MAP_H
00022  #define AAX_MAP_H
00023
00024  #include "AAX.h"
00025
00026  #include AAX_ALIGN_FILE_BEGIN
00027  #include AAX_ALIGN_FILE_ALG
00028  #include AAX_ALIGN_FILE_END
00029
00030  //=====
00031  class AAX_Map
00032  {
00033  public:
00034      AAX_Map() {};
00035      ~AAX_Map() {};
00036
00037      void SetCoefficients(int aSize, double* aInpX, double* aInpY);
00038      void GetCoefficient(int aIndex, double* aOutX, double* aOutY);
00039      int GetUpperBoundIndex(double inp);
00040      inline double GetX(int aIndex) {return mX[aIndex];};
00041      inline double GetY(int aIndex) {return mY[aIndex];};
00042      inline double GetFirstX() {return mX[0];};
00043      inline double GetFirstY() {return mY[0];};
00044      inline double GetLastX() {return mX[mSize - 1];};
00045      inline double GetLastY() {return mY[mSize - 1];};
00046      inline int GetSize() {return mSize;};
00047
00048  private:
00049
00050      static const int mMaxSize = 13;
00051
00052      int mSize;
00053
00054      double mX[mMaxSize];
00055      double mY[mMaxSize];
00056  };
00057
00058  #include AAX_ALIGN_FILE_BEGIN
00059  #include AAX_ALIGN_FILE_RESET
00060  #include AAX_ALIGN_FILE_END
00061
00062  #endif //AAX_MAP_H

```

15.338 AAX_MiscUtils.h File Reference

```

#include "AAX_PlatformOptimizationConstants.h"
#include "AAX_Constants.h"
#include "AAX_Denormal.h"

```

15.338.1 Description

Miscellaneous signal processing utilities.

Namespaces

- namespace [AAX](#)

Macros

- #define [AAX_MISCUTILS_H](#)
- #define [AAX_ALIGNMENT_HINT](#)(a, b)
Currently only functional on TI, these word alignments will provide better performance on TI.
- #define [AAX_WORD_ALIGNED_HINT](#)(a)
- #define [AAX_DWORD_ALIGNED_HINT](#)(a)
- #define [AAX_LO](#)(x) x
These macros are used on TI to convert 2 single words accesses to one double word access to provide additional optimization.
- #define [AAX_HI](#)(x) *((const_cast<float*>(&x))+1)
- #define [AAX_INT_LO](#)(x) x
- #define [AAX_INT_HI](#)(x) *((const_cast<int32_t*>(reinterpret_cast<const int32_t*>(&x)))+1)

Functions

- template<class GFLOAT >
GFLOAT [AAX::ClampToZero](#) (GFLOAT iValue, GFLOAT iClampThreshold)
- void [AAX::ZeroMemorySW](#) (void *iPointer, int iNumBytes)
- void [AAX::ZeroMemoryDW](#) (void *iPointer, int iNumBytes)
- template<typename T, int N>
void [AAX::Fill](#) (T *iArray, const T *iVal)
- template<typename T, int M, int N>
void [AAX::Fill](#) (T *iArray, const T *iVal)
- template<typename T, int L, int M, int N>
void [AAX::Fill](#) (T *iArray, const T *iVal)
- double [AAX::fabs](#) (double iVal)
- float [AAX::fabs](#) (float iVal)
- float [AAX::fabsf](#) (float iVal)
- template<class T >
T [AAX::AbsMax](#) (const T &iValue, const T &iMax)
- template<class T >
T [AAX::MinMax](#) (const T &iValue, const T &iMin, const T &iMax)
- template<class T >
T [AAX::Max](#) (const T &iValue1, const T &iValue2)
- template<class T >
T [AAX::Min](#) (const T &iValue1, const T &iValue2)
- template<class T >
T [AAX::Sign](#) (const T &iValue)
- double [AAX::PolyEval](#) (double x, const double *coefs, int numCoefs)
- double [AAX::CeilLog2](#) (double iValue)
- void [AAX::SinCosMix](#) (float aLinearMix, float &aSinMix, float &aCosMix)

15.338.2 Macro Definition Documentation

15.338.2.1 AAX_MISCUTILS_H

```
#define AAX_MISCUTILS_H
```

15.338.2.2 AAX_ALIGNMENT_HINT

```
#define AAX_ALIGNMENT_HINT(  
    a,  
    b )
```

Currently only functional on TI, these word alignments will provide better performance on TI.

15.338.2.3 AAX_WORD_ALIGNED_HINT

```
#define AAX_WORD_ALIGNED_HINT(  
    a )
```

15.338.2.4 AAX_DWORD_ALIGNED_HINT

```
#define AAX_DWORD_ALIGNED_HINT(  
    a )
```

15.338.2.5 AAX_LO

```
#define AAX_LO(  
    x ) x
```

These macros are used on TI to convert 2 single words accesses to one double word access to provide additional optimization.

15.338.2.6 AAX_HI

```
#define AAX_HI(  
    x ) *((const_cast<float*>(&x))+1)
```

15.338.2.7 AAX_INT_LO

```
#define AAX_INT_LO(
    x ) x
```

15.338.2.8 AAX_INT_HI

```
#define AAX_INT_HI(
    x ) *((const_cast<int32_t*>(reinterpret_cast<const int32_t*>(&x)))+1)
```

15.339 AAX_MiscUtils.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2013-2015, 2018, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00018 #pragma once
00019
00020 #ifndef AAX_MISCUTILS_H
00021 #define AAX_MISCUTILS_H
00022
00023 #include "AAX_PlatformOptimizationConstants.h"
00024 #include "AAX_Constants.h"
00025
00026 #if defined(_MSC_VER)
00027     #define DECLARE_ALIGNED(t,v,x)    __declspec(align(x)) t v
00028 // #define DECLARE_ALIGNED(t,v,x)    t v
00029 #elif defined (__GNUC__)
00030     #define DECLARE_ALIGNED(t,v,x)    t v __attribute__ ((aligned (x)))
00031 // #define DECLARE_ALIGNED(t,v,x)    t v
00032 #elif defined (_TMS320C6X)
00033     #define DECLARE_ALIGNED(t,v,x)    t v
00034     #warn "DECLARE_ALIGNED macro does not currently align data on TI"
00035 #else
00036     #error "DigiError: Please port the DECLARE_ALIGNED macro to this platform"
00037 #endif
00038
00039
00045 #ifndef _TMS320C6X
00046 #define AAX_ALIGNMENT_HINT(a,b)      std::_nassert(((int)(a) % b) == 0)
00047 #define AAX_WORD_ALIGNED_HINT(a)      AAX_ALIGNMENT_HINT(a,4)
00048 #define AAX_DWORD_ALIGNED_HINT(a)      AAX_ALIGNMENT_HINT(a,8)
00049 #else
00050 #define AAX_ALIGNMENT_HINT(a,b)
00051 #define AAX_WORD_ALIGNED_HINT(a)
00052 #define AAX_DWORD_ALIGNED_HINT(a)
00053 #endif
00054
00060 #ifndef _TMS320C6X
00061 #define AAX_LO(x)                    (_itof(_lo((_amemd8_const(&x))))))
00062 #define AAX_HI(x)                    (_itof(_hi((_amemd8_const(&x))))))
00063 #define AAX_INT_LO(x)                (_lo((_amemd8_const(&x))))
00064 #define AAX_INT_HI(x)                (_hi((_amemd8_const(&x))))
00065 #else //for non-TI systems increment pointer by 4-bytes to simulate hi-word access
00066 #define AAX_LO(x)                    x
00067 #define AAX_HI(x)                    *((const_cast<float*>(&x))+1)
00068 #define AAX_INT_LO(x)                x
00069 #define AAX_INT_HI(x)                *((const_cast<int32_t*>(reinterpret_cast<const int32_t*>(&x)))+1)
00070 #endif
00071
00072
00073
00074
```

```

00075 namespace AAX
00076 {
00077
00078 template<class GFLOAT>
00079 inline GFLOAT ClampToZero(GFLOAT iValue, GFLOAT iClampThreshold)
00080 {
00081     return (iValue < iClampThreshold && iValue > -iClampThreshold) ? 0.0 : iValue;
00082 }
00083
00084 /*template<class GFLOAT>
00085 class SmoothingFilter
00086 {
00087 public:
00088     SmoothingFilter() { Reset(); }
00089     void Reset(void) { mYnml=0.0; }
00090     void Reset(double iInitialState) { mYnml=iInitialState; }
00091     void SetSmoothingRate(double iFrequency, double iSampleRate) { mZeroCoef =
00092 1.0-std::exp(iFrequency*-cTwoPi/iSampleRate); }
00093     void SetSmoothingCoeff(GFLOAT b0) { mZeroCoef = b0; }
00094     inline GFLOAT Smooth(GFLOAT iInput)
00095     {
00096         mYnml += mZeroCoef * (iInput - mYnml);
00097         DeDenormal(mYnml);
00098         return (GFLOAT)mYnml;
00099     }
00100 private:
00101     double mYnml; // y[n-1]
00102     GFLOAT mZeroCoef;
00103 };*/
00104
00105 inline void ZeroMemorySW(void* iPointer, int iNumBytes)
00106 {
00107     char* aCharPointer=static_cast<char*>(iPointer);
00108
00109     for(int aIndex=0; aIndex<iNumBytes; aIndex++)
00110     {
00111         *aCharPointer=0;
00112         aCharPointer++;
00113     }
00114 }
00115
00116 //Warning: the number of bytes passed in must be multiple of 4
00117 inline void ZeroMemoryDW(void* iPointer, int iNumBytes)
00118 {
00119     int* aDWPointer=static_cast<int*>(iPointer);
00120
00121     int numDWords=iNumBytes/sizeof(int);
00122     for(int aIndex=0; aIndex<numDWords; aIndex++)
00123     {
00124         *aDWPointer=0;
00125         aDWPointer++;
00126     }
00127 }
00128
00129 template<typename T, int N> void Fill( T *iArray, const T* iVal )
00130 {
00131     // std::fill_n( iArray, N, iVal );
00132     for (int i = 0; i != N; ++i)
00133     {
00134         *(iArray + i) = *iVal;
00135     }
00136 }
00137
00138 template< typename T, int M, int N > inline void Fill( T *iArray, const T* iVal )//Fill( T
(&iArray)[M][N], const T& iVal )
00139 {
00140     for ( int i = 0; i != M; ++i )
00141     {
00142         Fill( iArray + i, iVal );
00143     }
00144 }
00145
00146 template< typename T, int L, int M, int N > inline void Fill( T *iArray, const T* iVal ) //Fill( T
(&iArray)[L][M][N], const T& iVal )
00147 {
00148     for ( int i = 0; i != L; ++i )
00149     {
00150         Fill( iArray + i, iVal );
00151     }
00152 }
00153
00154 static const int cSignBitWord = cLittleEndian;
00155
00156 double
00157 inline fabs(double iVal)
00158 {

```



```

00159 //On Windows this version is slower than std::fabs so use that instead.
00160 #if defined(MAC_VERSION)
00161 double aVal = iVal;
00162 int* tempPtr = (reinterpret_cast<int*>(&aVal))+cSignBitWord;
00163 *tempPtr &= 0x7fffffff;
00164
00165 return aVal;
00166 #else
00167 return std::fabs(iVal);
00168 #endif
00169 }
00170
00171 float
00172 inline fabs(float iVal)
00173 {
00174     // Call intrinsic if on TI c6727. fabs is about the same speed as std::fabs,
00175     // although metrowerks v9.5 MSL libraries are much slower so for the time
00176     // being were using this.
00177     int temp = (*(reinterpret_cast<int*>(&iVal)) & 0x7fffffff);
00178     return *reinterpret_cast<float*>(&temp);
00179 }
00180
00181 float
00182 inline fabsf(float iVal)
00183 {
00184     return AAX::fabs (iVal);
00185 }
00186
00187 template <class T>
00188 inline T AbsMax(const T& iValue, const T& iMax)
00189 {
00190     return std::fabs(iValue) < std::fabs(iMax) ? iMax : iValue;
00191 }
00192
00193 template <class T>
00194 inline T MinMax(const T& iValue, const T& iMin, const T& iMax)
00195 {
00196     return iValue > iMax ? iMax : (iValue < iMin ? iMin : iValue);
00197 }
00198
00199 template <class T>
00200 inline T Max(const T& iValue1, const T& iValue2)
00201 {
00202     return iValue1 > iValue2 ? iValue1 : iValue2;
00203 }
00204
00205 template <class T>
00206 inline T Min(const T& iValue1, const T& iValue2)
00207 {
00208     return iValue1 < iValue2 ? iValue1 : iValue2;
00209 }
00210
00211 template <class T>
00212 inline T Sign(const T& iValue)
00213 {
00214     return iValue < (T)(0.0) ? (T)(-1.0) : (T)(1.0);
00215 }
00216
00217 //Polynomial evaluation.
00218 inline double PolyEval(double x, const double* coefs, int numCoefs)
00219 {
00220     // Coefs are ordered from highest degree to lowest (Matlab convention)
00221     if(numCoefs < 1) return 0.0;
00222
00223     double result = coefs[0];
00224     for(int i = 1; i < numCoefs; ++i)
00225     {
00226         result = result*x + coefs[i];
00227     };
00228
00229     return result;
00230 }
00231
00232 inline double CeilLog2(double iValue)
00233 {
00234     return std::pow(2.0f, (float)(std::ceil(std::log(iValue)/std::log(2.0f))));
00235 }
00236
00237 inline void SinCosMix(float aLinearMix, float &aSinMix, float &aCosMix)
00238 {
00239     aSinMix=static_cast< float >( std::sin(aLinearMix*cHalfPi) );
00240     aCosMix=static_cast< float >( std::cos(aLinearMix*cHalfPi) );
00241 }
00242
00243
00244 } // namespace AAX
00245

```

```

00246
00247 // Some methods have been broken off into AAX_Denormal.h; including
00248 // AAX_Denormal.h here for compatibility with projects that have not
00249 // yet been updated to include this new header.
00250 #include "AAX_Denormal.h"
00251
00252 #endif // AAX_MISCTILS_H

```

15.340 AAX_PlatformOptimizationConstants.h File Reference

15.340.1 Description

Constants descriptor...

Macros

- `#define AAX_PLATFORMOPTIMIZATIONCONSTANTS_H`

15.340.2 Macro Definition Documentation

15.340.2.1 AAX_PLATFORMOPTIMIZATIONCONSTANTS_H

```
#define AAX_PLATFORMOPTIMIZATIONCONSTANTS_H
```

15.341 AAX_PlatformOptimizationConstants.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2009-2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00018 #pragma once
00019
00020 #ifndef AAX_PLATFORMOPTIMIZATIONCONSTANTS_H
00021 #define AAX_PLATFORMOPTIMIZATIONCONSTANTS_H
00022
00023 // Set up our platform-specific optimization defines
00024 #if USE_PLATFORM_OPTIMIZATION
00025     #if defined( WINDOWS_VERSION )
00026         #define USE_INTEL_IPP (1)           // Windows
00027         #define __SSE__ (1)                 // Manually define the __SSE__ flag
00028     #elif defined( MAC_VERSION )
00029         #if defined( __ppc__ )
00030             #define USE_ALTIVEC_VDSP (1)    // PPC
00031         #elif defined( __i386__ ) or defined( __x86_64__ )
00032             #define USE_INTEL_IPP (1)       // MacTel
00033         #else
00034             #error "Unsupported platform for optimizations!"
00035         #endif // __i386__ or __ppc__
00036     #else
00037         #error "Unsupported platform for optimizations!"
00038     #endif // WINDOWS_VERSION
00039 #endif // USE_PLATFORM_OPTIMIZATION
00040
00041 #endif // AAX_PLATFORMOPTIMIZATIONCONSTANTS_H
00042

```

15.342 AAX_Quantize.h File Reference

```
#include "AAX.h"
#include "AAX_PlatformOptimizationConstants.h"
#include "AAX_Constants.h"
#include <xmmintrin.h>
#include <pmmmintrin.h>
#include <tmmmintrin.h>
```

15.342.1 Description

Quantization utilities.

Namespaces

- namespace [AAX](#)

Macros

- #define [AAX_QUANTIZE_H](#)

Functions

- `int32_t AAX::FastRound2Int32 (double iVal)`
Round to Int32.
- `int32_t AAX::FastRound2Int32 (float iVal)`
Round to Int32.
- `int32_t AAX::FastRndDbl2Int32 (double iVal)`
- `int32_t AAX::FastTrunc2Int32 (double iVal)`
Float to Int conversion with truncation.
- `int32_t AAX::FastTrunc2Int32 (float iVal)`
Float to Int conversion with truncation.
- `int64_t AAX::FastRound2Int64 (double iVal)`
Round to Int64.

15.342.2 Macro Definition Documentation

15.342.2.1 AAX_QUANTIZE_H

```
#define AAX_QUANTIZE_H
```

15.343 AAX_Quantize.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2013-2015, 2019, 2021, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00018 #pragma once
00019
00020 #ifndef AAX_QUANTIZE_H
00021 #define AAX_QUANTIZE_H
00022
00023 #include "AAX.h"
00024 #include "AAX_PlatformOptimizationConstants.h"
00025 #include "AAX_Constants.h"
00026
00027 #if ! defined( _TMS320C6X )
00028
00029 #if _MSC_VER && !defined(__INTEL_COMPILER)
00030 #define _MM_FUNCTIONALITY
00031 #include <intrin.h>
00032 #elif LINUX_VERSION
00033 #elif TARGET_OS_IPHONE
00034 #elif defined(__arm__)
00035 #elif defined(__arm64__)
00036 #else
00037 // GNU or INTEL:
00038 #include <xmmintrin.h>
00039 #include <pmmintrin.h>
00040 #include <tmmintrin.h>
00041 #endif
00042
00043 #endif
00044
00045 namespace AAX
00046 {
00047
00048 //This assumes the floating point processor is in 64 bit IEEE mode.
00049 #if ! defined( _TMS320C6X )
00050
00051 // For double->Int32 conversion
00052 static const double cDouble2IntBias = ldexpf(1,52)*1.5;
00053 static const double cOneHalfOffset = 0.5;
00054 static const int32_t cMantisaWord = cBigEndian;
00055
00056 // For double->Int64 conversion
00057 static const double kExponentMagicDelta = 1.5e-8; //1e^(number of
exp bit)
00058 static const double kBigMantissaMagicFloat = 6755399441055744.0; //2^52 * 1.5,
uses limited precision to floor
00059 static const int64_t kBigMantissaMagicMask = 0x1fffffffffffffLL; //2^52 *
1.5 mask,
00060 static const int64_t kBigMantissaMagicInt = 0x18000000000000LL; //2^52 *
1.5, uses limited precision to floor
00061 #endif
00062
00063 /*=====*/
00064
00073 inline int32_t FastRound2Int32(double iVal)
00074 {
00075 #if defined( _TMS320C6X )
00076 // rounding mode set by the CSR register
00077 const int32_t r = _drint(iVal);
00078 return r;
00079 #else
00080 iVal += cDouble2IntBias;
00081 return (reinterpret_cast<int32_t*>(&iVal))[cMantisaWord];
00082 #endif
00083 }
00084
00093 inline int32_t FastRound2Int32(float iVal)
00094 {
00095 #if defined( _TMS320C6X )
00096 // rounding mode set by the CSR register
00097 const int32_t r = _srint(iVal);
00098 return r;
00099 #else

```

```

00100     return FastRound2Int32(double(iVal));
00101 #endif
00102 }
00103
00104
00107 inline int32_t FastRndDb12Int32(double iVal)
00108 {
00109     return FastRound2Int32(iVal);
00110 }
00111
00124 inline int32_t FastTrunc2Int32(double iVal)
00125 {
00126     #if defined( _TMS320C6X )
00127         // rounding mode set by the CSR register
00128         const int32_t r = _dpint(iVal - 0.5);
00129         return r;
00130     #else
00131         return FastRound2Int32(iVal - 0.5);
00132     #endif
00133 }
00134
00143 inline int32_t FastTrunc2Int32(float iVal)
00144 {
00145     #if defined( _TMS320C6X )
00146         // rounding mode set by the CSR register
00147         const int32_t r = _spint(iVal - 0.5f);
00148         return r;
00149     #elif _MSC_VER
00150         int32_t r;
00151         r = _mm_cvtt_ss2si( _mm_load_ss(&iVal) );
00152         return r;
00153     #else
00154         return static_cast<int32_t>(iVal);
00155     #endif
00156 }
00157
00167 inline int64_t FastRound2Int64(double iVal)
00168 {
00169     #if defined( _TMS320C6X )
00170         return (int64_t)(iVal > 0.0 ? iVal + 0.5 : iVal - 0.5);
00171     #else
00172         iVal += kExponentMagicDelta; // Round to nearest
00173         iVal += kBigMantissaMagicFloat; // Normalize to integer
00174         int64_t result = *reinterpret_cast<int64_t*>(&iVal);
00175         result &= kBigMantissaMagicMask; // Mask out upper bits of float (exponent, sign)
00176         result -= kBigMantissaMagicInt; // Normalize to integer
00177         return result;
00178     #endif
00179 }
00180
00181 } // namespace AAX
00182
00183 #endif // AAX_QUANTIZE_H

```

15.344 AAX_RandomGen.h File Reference

```

#include <stdlib.h>
#include <time.h>
#include <stdint.h>
#include "AAX_PlatformOptimizationConstants.h"
#include "AAX_Constants.h"

```

15.344.1 Description

Functions for calculating pseudo-random numbers.

Namespaces

- namespace [AAX](#)

Macros

- `#define AAX_RANDOMGEN_H`

Functions

- `int32_t AAX::GetInt32RPDF (int32_t *iSeed)`
- `int32_t AAX::GetFastInt32RPDF (int32_t *iSeed)`
CALL: Calculate pseudo-random 32 bit number based on linear congruential method.
- `float AAX::GetRPDFWithAmplitudeOneHalf (int32_t *iSeed)`
- `float AAX::GetRPDFWithAmplitudeOne (int32_t *iSeed)`
- `float AAX::GetFastRPDFWithAmplitudeOne (int32_t *iSeed)`
- `float AAX::GetTPDFWithAmplitudeOne (int32_t *iSeed)`

Variables

- `const float AAX::cSeedDivisor = 1/127773.0f`
- `const int32_t AAX::cInitialSeedValue =0x00F54321`

15.344.2 Macro Definition Documentation

15.344.2.1 AAX_RANDOMGEN_H

```
#define AAX_RANDOMGEN_H
```

15.345 AAX_RandomGen.h

[Go to the documentation of this file.](#)

```
00001 /*=====*/
00002 /*
00003  * Copyright 2013-2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00018 #pragma once
00019
00020 #ifndef AAX_RANDOMGEN_H
00021 #define AAX_RANDOMGEN_H
00022
00023 // Standard headers
00024 #include <stdlib.h>
00025 #include <time.h>
00026 #include <stdint.h>
00027
00028 #include "AAX_PlatformOptimizationConstants.h"
00029 #include "AAX_Constants.h"
00030
00031 namespace AAX
00032 {
00033
00034     const float cSeedDivisor = 1/127773.0f;
```

```

00035
00036 const int32_t cInitialSeedValue=0x00F54321;
00037
00038
00039 /*=====*/
00039 inline int32_t GetInt32RPDF(int32_t* iSeed)
00040 {
00041     // Requirement: iSeed param must be a static in the calling function.
00042     int64_t seed = *iSeed;
00043     int64_t k = static_cast<int64_t>(seed*cSeedDivisor);
00044     seed = 16807 * (seed - k * 127773LL) - 2836 * k + 7395;
00045     *iSeed = static_cast<int32_t>(seed);
00046     if (*iSeed < 0)
00047         *iSeed += 2147483647;
00048     return (*iSeed - 1073741824) * 2;          // -2147483647..+2147483647
00049 }
00050
00051 /*=====*/
00052
00053
00066 inline int32_t GetFastInt32RPDF(int32_t* iSeed)
00067 {
00068     *iSeed = (*iSeed * 196314165) + 907633515;
00069     return *iSeed;
00070 }
00071
00072 /*=====*/
00073 inline float GetRPDFWithAmplitudeOneHalf(int32_t* iSeed)    //An amplitude of 0.5 = 1 LSBs
00074 {
00075     // Requirement: iSeed param must be a static in the calling function.
00076     return float(cNormalizeLongToAmplitudeOneHalf) * float(GetInt32RPDF(iSeed));
00077 }
00078
00079 /*=====*/
00080 inline float GetRPDFWithAmplitudeOne(int32_t* iSeed)        //An amplitude of 1.0 = 2 LSBs
00081 {
00082     // Requirement: iSeed param must be a static in the calling function.
00083     return float(cNormalizeLongToAmplitudeOne) * float(GetInt32RPDF(iSeed));
00084 }
00085
00086 /*=====*/
00087 inline float GetFastRPDFWithAmplitudeOne(int32_t* iSeed)    //An amplitude of 1.0 = 2 LSBs
00088 {
00089     // Requirement: iSeed param must be a static in the calling function.
00090     return float(cNormalizeLongToAmplitudeOne) * float(GetFastInt32RPDF(iSeed));
00091 }
00092
00093 /*=====*/
00094 inline float GetTPDFWithAmplitudeOne(int32_t* iSeed)        //An amplitude of 1.0 = 2 LSBs
00095 {
00096     // Generate a random number with a triangular pdf (tpdf) using two
00097     // separate rectangular pdf noise sources.
00098
00099     // We are using only one seed input for both rect pdf noise generators,
00100     // but because they are of course executed sequentially independent noise
00101     // will be produced.
00102
00103     return float(cNormalizeLongToAmplitudeOne) * (float(GetFastInt32RPDF(iSeed) +
00104         float(GetFastInt32RPDF(iSeed))));
00105 }
00106 } // namespace AAX
00107
00108 #endif // AAX_RANDOMGEN_H
00109

```

15.346 AAX_SampleRateUtils.h File Reference

15.346.1 Description

Description.

Enumerations

- enum [ESRUtils](#) {
 - [eSRUtils_48kRangeCoarse](#) = 48000 ,
 - [eSRUtils_96kRangeCoarse](#) = 96000 ,
 - [eSRUtils_192kRangeCoarse](#) = 192000 ,
 - [eSRUtils_48kRangeMin](#) = 0 ,
 - [eSRUtils_48kRangeMax](#) = 51000 ,
 - [eSRUtils_96kRangeMin](#) = [eSRUtils_48kRangeMax](#)+1 ,
 - [eSRUtils_96kRangeMax](#) = 102000 ,
 - [eSRUtils_192kRangeMin](#) = [eSRUtils_96kRangeMax](#)+1 ,
 - [eSRUtils_192kRangeMax](#) = 204000 ,
 - [eSRUtils_48kIndex](#) = 0 ,
 - [eSRUtils_96kIndex](#) = 1 ,
 - [eSRUtils_192kIndex](#) = 2 }

Functions

- int [CoarseSampleRate](#) (int iRate)
- int [CoarseSampleRateFactor](#) (int iRate)
- int [CoarseSampleRateIndex](#) (int iRate)

15.346.2 Enumeration Type Documentation

15.346.2.1 ESRUtils

enum [ESRUtils](#)

Enumerator

eSRUtils_48kRangeCoarse	
eSRUtils_96kRangeCoarse	
eSRUtils_192kRangeCoarse	
eSRUtils_48kRangeMin	
eSRUtils_48kRangeMax	
eSRUtils_96kRangeMin	
eSRUtils_96kRangeMax	
eSRUtils_192kRangeMin	
eSRUtils_192kRangeMax	
eSRUtils_48kIndex	
eSRUtils_96kIndex	
eSRUtils_192kIndex	

15.346.3 Function Documentation

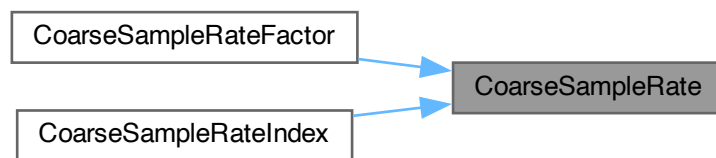
15.346.3.1 CoarseSampleRate()

```
int CoarseSampleRate (  
    int iRate ) [inline]
```

References [eSRUtils_192kRangeCoarse](#), [eSRUtils_192kRangeMax](#), [eSRUtils_192kRangeMin](#), [eSRUtils_48kRangeCoarse](#), [eSRUtils_48kRangeMax](#), [eSRUtils_48kRangeMin](#), [eSRUtils_96kRangeCoarse](#), [eSRUtils_96kRangeMax](#), and [eSRUtils_96kRangeMin](#).

Referenced by [CoarseSampleRateFactor\(\)](#), and [CoarseSampleRateIndex\(\)](#).

Here is the caller graph for this function:



15.346.3.2 CoarseSampleRateFactor()

```
int CoarseSampleRateFactor (  
    int iRate ) [inline]
```

References [CoarseSampleRate\(\)](#), and [eSRUtils_48kRangeCoarse](#).

Here is the call graph for this function:



15.346.3.3 CoarseSampleRateIndex()

```
int CoarseSampleRateIndex (
    int iRate ) [inline]
```

References [CoarseSampleRate\(\)](#), [eSRUtils_192kRangeCoarse](#), [eSRUtils_48kRangeCoarse](#), and [eSRUtils_96kRangeCoarse](#).

Here is the call graph for this function:



15.347 AAX_SampleRateUtils.h

[Go to the documentation of this file.](#)

```

00001 /*=====*/
00002 /*
00003  * Copyright 2009–2015, 2023 Avid Technology, Inc.
00004  * All rights reserved.
00005  *
00006  * CONFIDENTIAL: this document contains confidential information of Avid. Do
00007  * not disclose to any third party. Use of the information contained in this
00008  * document is subject to an Avid SDK license.
00009  */
00010
00017 /*=====*/
00018 #pragma once
00019
00020 enum ESRUtils
00021 {
00022     eSRUtils_48kRangeCoarse      = 48000,
00023     eSRUtils_96kRangeCoarse      = 96000,
00024     eSRUtils_192kRangeCoarse     = 192000,
00025     eSRUtils_48kRangeMin         = 0,
00026     eSRUtils_48kRangeMax         = 51000,
00027     eSRUtils_96kRangeMin         = eSRUtils_48kRangeMax+1,
00028     eSRUtils_96kRangeMax         = 102000,
00029     eSRUtils_192kRangeMin        = eSRUtils_96kRangeMax+1,
00030     eSRUtils_192kRangeMax        = 204000,
00031     eSRUtils_48kIndex            = 0,
00032     eSRUtils_96kIndex            = 1,
00033     eSRUtils_192kIndex           = 2
00034 };
00035
00036 inline int CoarseSampleRate (int iRate)
00037 {
00038     const int aCoarseRate =
00039
00040         ((iRate >= eSRUtils_48kRangeMin) && (iRate <= eSRUtils_48kRangeMax)) ?
00041         eSRUtils_48kRangeCoarse :
00042         ((iRate >= eSRUtils_96kRangeMin) && (iRate <= eSRUtils_96kRangeMax)) ?
00043         eSRUtils_96kRangeCoarse :
00044         ((iRate >= eSRUtils_192kRangeMin) && (iRate <= eSRUtils_192kRangeMax)) ?
00045         eSRUtils_192kRangeCoarse :
00046         0;
00047
00048     if (aCoarseRate == 0)
00049     {
00050         throw std::runtime_error ("unrecognized sample rate");
00051     }
00052     return aCoarseRate;
00053 }
00054
00055 //Returns 1 for 48k, 2 for 96k, and 4 for 192k gross samples rate.
```

```
00053 inline int CoarseSampleRateFactor (int iRate)
00054 {
00055     const int kMinCoarseSampleRate=eSRUtils_48kRangeCoarse;
00056
00057     int aCoarseRateFactor = CoarseSampleRate (iRate)/kMinCoarseSampleRate;
00058
00059     return aCoarseRateFactor;
00060 }
00061
00062 //Returns 0 for 48k, 1 for 96k, and 2 for 192k gross samples rate.
00063 inline int CoarseSampleRateIndex (int iRate)
00064 {
00065     // const long kMinGrossSampleRate=eSRUtils_48kRangeCoarse;
00066     int aGrossRateIndex = 0;
00067
00068     switch ( CoarseSampleRate (iRate) )
00069     {
00070     default:
00071         case eSRUtils_48kRangeCoarse:
00072             aGrossRateIndex = 0;
00073             break;
00074         case eSRUtils_96kRangeCoarse:
00075             aGrossRateIndex = 1;
00076             break;
00077         case eSRUtils_192kRangeCoarse:
00078             aGrossRateIndex = 2;
00079             break;
00080         // default:
00081         //     throw std::runtime_error ("unrecognized sample rate");
00082     }
00083
00084     return aGrossRateIndex;
00085 }
```


Index

- [_AAX_CAUTORELEASEPOOL_H_](#)
 - [AAX_CAutoreleasePool.h, 1350](#)
 - [_AAX_FASTPOW_H_](#)
 - [AAX_FastPow.h, 1711](#)
 - [_AAX_UTILSNATIVE_H_](#)
 - [AAX_UtillsNative.h, 1672](#)
 - [_AAX_VERSION_H_](#)
 - [AAX_Version.h, 1683](#)
 - [_acfUID, 411](#)
 - [Data1, 411](#)
 - [Data2, 411](#)
 - [Data3, 411](#)
 - [Data4, 411](#)
- [~AAX_AggregateResult](#)
- [AAX_AggregateResult, 413](#)
- [~AAX_CArrayDataBuffer](#)
- [AAX_CArrayDataBuffer< D >, 419](#)
- [~AAX_CArrayDataBufferOfType](#)
- [AAX_CArrayDataBufferOfType< T, D >, 423](#)
- [~AAX_CAtomicQueue](#)
- [AAX_CAtomicQueue< T, S >, 427](#)
- [~AAX_CAutoreleasePool](#)
- [AAX_CAutoreleasePool, 430](#)
- [~AAX_CChunkDataParser](#)
- [AAX_CChunkDataParser, 441](#)
- [~AAX_CEffectDirectData](#)
- [AAX_CEffectDirectData, 455](#)
- [~AAX_CEffectGUI](#)
- [AAX_CEffectGUI, 462](#)
- [~AAX_CEffectParameters](#)
- [AAX_CEffectParameters, 478](#)
- [~AAX_CHostProcessor](#)
- [AAX_CHostProcessor, 516](#)
- [~AAX_CMonolithicParameters](#)
- [AAX_CMonolithicParameters, 550](#)
- [~AAX_CMutex](#)
- [AAX_CMutex, 559](#)
- [~AAX_CPacket](#)
- [AAX_CPacket, 567](#)
- [~AAX_CPacketDispatcher](#)
- [AAX_CPacketDispatcher, 569](#)
- [~AAX_CParameter](#)
- [AAX_CParameter< T >, 584](#)
- [~AAX_CParameterManager](#)
- [AAX_CParameterManager, 616](#)
- [~AAX_CPieceWiseLinearTaperDelegate](#)
- [AAX_CPieceWiseLinearTaperDelegate< T, Real-Precision >, 638](#)
- [~AAX_CSessionDocumentClient](#)
- [AAX_CSessionDocumentClient, 650](#)
- [~AAX_CStringDataBuffer](#)
- [AAX_CStringDataBuffer, 712](#)
- [~AAX_CStringDataBufferOfType](#)
- [AAX_CStringDataBufferOfType< T >, 716](#)
- [~AAX_CTaskAgent](#)
- [AAX_CTaskAgent, 725](#)
- [~AAX_CheckedResult](#)
- [AAX_CheckedResult, 509](#)
- [~AAX_IAutomationDelegate](#)
- [AAX_IAutomationDelegate, 941](#)
- [~AAX_ICollection](#)
- [AAX_ICollection, 947](#)
- [~AAX_IComponentDescriptor](#)
- [AAX_IComponentDescriptor, 953](#)
- [~AAX_IContainer](#)
- [AAX_IContainer, 969](#)
- [~AAX_IController](#)
- [AAX_IController, 972](#)
- [~AAX_IDataBufferWrapper](#)
- [AAX_IDataBufferWrapper, 989](#)
- [~AAX_IDescriptionHost](#)
- [AAX_IDescriptionHost, 990](#)
- [~AAX_IDisplayDelegateBase](#)
- [AAX_IDisplayDelegateBase, 997](#)
- [~AAX_IDisplayDelegateDecorator](#)
- [AAX_IDisplayDelegateDecorator< T >, 1000](#)
- [~AAX_IDma](#)
- [AAX_IDma, 1007](#)
- [~AAX_IEffectDescriptor](#)
- [AAX_IEffectDescriptor, 1015](#)
- [~AAX_IFeatureInfo](#)
- [AAX_IFeatureInfo, 1034](#)
- [~AAX_IHostProcessorDelegate](#)
- [AAX_IHostProcessorDelegate, 1039](#)
- [~AAX_IHostServices](#)
- [AAX_IHostServices, 1042](#)
- [~AAX_IMIDIMessageInfoDelegate](#)
- [AAX_IMIDIMessageInfoDelegate, 1044](#)
- [~AAX_IMIDINode](#)
- [AAX_IMIDINode, 1049](#)
- [~AAX_IPacketHandler](#)
- [AAX_IPacketHandler, 1051](#)
- [~AAX_IPageTable](#)
- [AAX_IPageTable, 1052](#)
- [~AAX_IParameter](#)
- [AAX_IParameter, 1066](#)
- [~AAX_IParameterValue](#)
- [AAX_IParameterValue, 1086](#)

- ~AAX_IPointerQueue
 - AAX_IPointerQueue< T >, 1091
- ~AAX_IPrivateDataAccess
 - AAX_IPrivateDataAccess, 1093
- ~AAX_IPropertyMap
 - AAX_IPropertyMap, 1096
- ~AAX_ISessionDocument
 - AAX_ISessionDocument, 1101
- ~AAX_IString
 - AAX_IString, 1105
- ~AAX_ITaperDelegateBase
 - AAX_ITaperDelegateBase, 1112
- ~AAX_ITask
 - AAX_ITask, 1113
- ~AAX_ITransport
 - AAX_ITransport, 1120
- ~AAX_IViewContainer
 - AAX_IViewContainer, 1129
- ~AAX_Map
 - AAX_Map, 1135
- ~AAX_StLock_Guard
 - AAX_StLock_Guard, 1160
- ~AAX_VAutomationDelegate
 - AAX_VAutomationDelegate, 1165
- ~AAX_VCollection
 - AAX_VCollection, 1170
- ~AAX_VComponentDescriptor
 - AAX_VComponentDescriptor, 1178
- ~AAX_VController
 - AAX_VController, 1192
- ~AAX_VDataBufferWrapper
 - AAX_VDataBufferWrapper, 1208
- ~AAX_VDescriptionHost
 - AAX_VDescriptionHost, 1211
- ~AAX_VEffectDescriptor
 - AAX_VEffectDescriptor, 1215
- ~AAX_VFeatureInfo
 - AAX_VFeatureInfo, 1220
- ~AAX_VHostServices
 - AAX_VHostServices, 1226
- ~AAX_VPageTable
 - AAX_VPageTable, 1231
- ~AAX_VPrivateDataAccess
 - AAX_VPrivateDataAccess, 1242
- ~AAX_VPropertyMap
 - AAX_VPropertyMap, 1245
- ~AAX_VSessionDocument
 - AAX_VSessionDocument, 1251
- ~AAX_VTask
 - AAX_VTask, 1254
- ~AAX_VTransport
 - AAX_VTransport, 1259
- ~AAX_VViewContainer
 - AAX_VViewContainer, 1268
- ~Any
 - AAX::Exception::Any, 1274
- ~SAutoArray
 - SAutoArray< T >, 1287
- ~TempoMap
 - AAX_ISessionDocument::TempoMap, 1289
- ~VTempoMap
 - AAX_VSessionDocument::VTempoMap, 1291
- AAE_EAudioBufferLengthNative
 - AAX_Enums.h, 1450
- AAX, 371
 - AbsMax, 395
 - alignFree, 390
 - alignMalloc, 391
 - AsString, 377
 - AsStringFourChar, 385
 - AsStringIDTriad, 387
 - AsStringInt32, 387
 - AsStringPropertyValue, 386
 - AsStringResult, 388
 - AsStringStemChannel, 388
 - AsStringStemFormat, 387
 - AsStringUInt32, 387
 - Binary2String, 383
 - Caseless_strcmp, 383
 - cBigEndian, 402
 - cDenormalAvoidanceOffset, 405
 - CeilLog2, 396
 - cFloatDenormalAvoidanceOffset, 405
 - cGiga, 405
 - cHalfPi, 403
 - cInitialSeedValue, 406
 - cKilo, 405
 - ClampToZero, 392
 - ClearMappedParameterByID, 382
 - cLittleEndian, 402
 - cMega, 405
 - cMicro, 404
 - cMilli, 404
 - cNano, 405
 - cNeg3dB, 404
 - cNeg6dB, 404
 - cNormalizeLongToAmplitudeOne, 404
 - cNormalizeLongToAmplitudeOneHalf, 404
 - cOneOverRootTwo, 403
 - CopyPageTable, 381
 - cPi, 403
 - cPico, 405
 - cPos3dB, 403
 - cPos6dB, 404
 - cQuarterPi, 403
 - cRootTwo, 403
 - cSeedDivisor, 406
 - cTwoPi, 403
 - DeDenormal, 391
 - DeDenormalFine, 391
 - e176400SampleRate, 377
 - e192000SampleRate, 377
 - e44100SampleRate, 377
 - e48000SampleRate, 377
 - e88200SampleRate, 377
 - e96000SampleRate, 377

- EChannelModeData, 376
- eChannelModeData_AllNotesOff, 376
- eChannelModeData_AllSoundOff, 376
- eChannelModeData_LocalControl, 376
- eChannelModeData_OmniOff, 376
- eChannelModeData_OmniOn, 376
- eChannelModeData_PolyOff, 376
- eChannelModeData_PolyOn, 376
- eChannelModeData_ResetControllers, 376
- ESampleRates, 377
- ESpecialData, 376
- eSpecialData_AccentedClick, 376
- eSpecialData_UnaccentedClick, 376
- EStatusByte, 375
- eStatusByte_ActiveSensing, 376
- eStatusByte_Continue, 376
- eStatusByte_MTCQuarterFrame, 376
- eStatusByte_Reset, 376
- eStatusByte_SongPosition, 376
- eStatusByte_SongSelect, 376
- eStatusByte_Start, 376
- eStatusByte_Stop, 376
- eStatusByte_SysExBegin, 375
- eStatusByte_SysExEnd, 376
- eStatusByte_TimingClock, 376
- eStatusByte_TuneRequest, 376
- EStatusNibble, 375
- eStatusNibble_ChannelMode, 375
- eStatusNibble_ChannelPressure, 375
- eStatusNibble_ControlChange, 375
- eStatusNibble_KeyPressure, 375
- eStatusNibble_NoteOff, 375
- eStatusNibble_NoteOn, 375
- eStatusNibble_PitchBend, 375
- eStatusNibble_ProgramChange, 375
- eStatusNibble_SystemCommon, 375
- eStatusNibble_SystemRealTime, 375
- fabs, 394
- fabsf, 394
- FastRndDbl2Int32, 397
- FastRound2Int32, 396, 397
- FastRound2Int64, 399
- FastTrunc2Int32, 398, 399
- Fill, 392, 393
- FilterDenormals, 391
- FindParameterMappingsInPageTable, 382
- GetCStringOfLength, 383
- GetFastInt32RPDF, 400
- GetFastRPDFWithAmplitudeOne, 401
- GetInt32RPDF, 400
- GetRPDFWithAmplitudeOne, 401
- GetRPDFWithAmplitudeOneHalf, 401
- GetTPDFWithAmplitudeOne, 402
- IsAccentedClick, 378
- IsAllNotesOff, 378
- IsASCII, 384
- IsAvidNotification, 390
- IsClick, 379
- IsEffectIDEqual, 390
- IsFourCharASCII, 385
- IsNoteOff, 378
- IsNoteOn, 378
- IsParameterIDEqual, 390
- IsUnaccentedClick, 379
- kPowExtent, 406
- kPowTableSize, 406
- Max, 395
- Min, 396
- MinMax, 395
- PageTableParameterMappingsAreEqual, 380
- PageTableParameterNameVariationsAreEqual, 380
- PageTablesAreEqual, 381
- PolyEval, 396
- SafeLog, 389
- SafeLogf, 389
- Sign, 396
- SinCosMix, 396
- String2Binary, 384
- ZeroMemoryDW, 392
- ZeroMemorySW, 392
- AAX communication protocols, 76
- AAX Format Specification, 78
- AAX Host Guides, 150
- AAX Interfaces, 294
- AAX Library features, 105
- AAX SDK Manual, 43
- AAX.h, 1303, 1319
 - AAX_ALIGN_FILE_ALG, 1309
 - AAX_ALIGN_FILE_BEGIN, 1310
 - AAX_ALIGN_FILE_END, 1310
 - AAX_ALIGN_FILE_HOST, 1309
 - AAX_ALIGN_FILE_RESET, 1310
 - AAX_CALLBACK, 1310
 - AAX_CAudioInPort, 1316
 - AAX_CAudioOutPort, 1316
 - AAX_CBoolean, 1312
 - AAX_CComponentID, 1314
 - AAX_CCount, 1312
 - AAX_CEffectID, 1315
 - AAX_CFieldIndex, 1314
 - AAX_CIndex, 1311
 - AAX_CMeterID, 1314
 - AAX_CMeterPort, 1316
 - AAX_CONSTEXPR, 1308
 - AAX_CPageTableParamID, 1315
 - AAX_CParamID, 1314
 - AAX_CPointerPropertyValue, 1313
 - AAX_CPP11_SUPPORT, 1306
 - AAX_CPropertyValue, 1313
 - AAX_CPropertyValue64, 1313
 - AAX_CSampleRate, 1313
 - AAX_CSelector, 1312
 - AAX_CTargetPlatform, 1314
 - AAX_CTimeOfDay, 1312
 - AAX_CTimestamp, 1312

- AAX_CTransportCounter, [1312](#)
- AAX_CTypeID, [1313](#)
- AAX_DEFAULT_ASGN_OPER, [1308](#)
- AAX_DEFAULT_COPY_CTOR, [1307](#)
- AAX_DEFAULT_CTOR, [1307](#)
- AAX_DEFAULT_DTOR, [1307](#)
- AAX_DEFAULT_DTOR_OVERRIDE, [1307](#)
- AAX_DEFAULT_MOVE_CTOR, [1307](#)
- AAX_DEFAULT_MOVE_OPER, [1308](#)
- AAX_DELETE, [1308](#)
- AAX_Feature_UID, [1315](#)
- AAX_FIELD_INDEX, [1311](#)
- AAX_FINAL, [1307](#)
- AAX_OVERRIDE, [1306](#)
- AAX_PointerSize, [1309](#)
- AAX_PREPROCESSOR_CONCAT, [1311](#)
- AAX_PREPROCESSOR_CONCAT_HELPER, [1311](#)
- AAX_Result, [1313](#)
- AAX_SPlugInChunk, [1317](#)
- AAX_SPlugInChunkHeader, [1316](#)
- AAX_SPlugInChunkPtr, [1317](#)
- AAX_SPlugInIdentifierTriad, [1317](#)
- AAX_SPlugInIdentifierTriadPtr, [1317](#)
- AAX_UNIQUE_PTR, [1308](#)
- AAXPointer_32bit, [1309](#)
- AAXPointer_64bit, [1309](#)
- acfUID, [1315](#)
- getLowestSampleRateInMask, [1317](#)
- getMaskForSampleRate, [1318](#)
- kAAX_ParameterIdentifierMaxSize, [1318](#)
- sampleRateInMask, [1317](#)
- TI_VERSION, [1306](#)
- AAX::Exception, [406](#)
- AAX::Exception::Any, [1273](#)
 - ~Any, [1274](#)
 - AAX_DEFAULT_MOVE_CTOR, [1275](#)
 - AAX_DEFAULT_MOVE_OPER, [1275](#)
 - Any, [1275](#)
 - Desc, [1276](#)
 - Function, [1276](#)
 - Line, [1276](#)
 - operator=, [1275](#)
 - What, [1275](#)
- AAX::Exception::ResultError, [1284](#)
 - FormatResult, [1286](#)
 - Result, [1286](#)
 - ResultError, [1286](#)
- AAX::internal, [407](#)
 - ToHexadecimal, [407](#)
- AAX_ACFInterface.doxygen, [1293](#)
 - acfIID, [1293](#)
 - acfUID, [1293](#)
- AAX_AdditionalFeatures_Algorithm.doxygen, [1294](#)
- AAX_AdditionalFeatures_AOSandSidechain.doxygen, [1294](#)
- AAX_AdditionalFeatures_CurveDisplays.doxygen, [1294](#)
- AAX_AdditionalFeatures_Hybrid.doxygen, [1294](#)
- AAX_AdditionalFeatures_Meters.doxygen, [1294](#)
- AAX_AdditionalFeatures_MIDI.doxygen, [1294](#)
- AAX_AggregateResult, [412](#)
 - ~AAX_AggregateResult, [413](#)
 - AAX_AggregateResult, [413](#)
 - Check, [414](#)
 - Clear, [414](#)
 - LastFailure, [415](#)
 - NumAttempted, [416](#)
 - NumFailed, [415](#)
 - NumSucceeded, [415](#)
 - operator AAX_Result, [413](#)
 - operator=, [413](#)
- AAX_ALIGN_FILE_ALG
 - AAX.h, [1309](#)
- AAX_ALIGN_FILE_BEGIN
 - AAX.h, [1310](#)
- AAX_ALIGN_FILE_END
 - AAX.h, [1310](#)
- AAX_ALIGN_FILE_HOST
 - AAX.h, [1309](#)
- AAX_ALIGN_FILE_RESET
 - AAX.h, [1310](#)
- AAX_Alignment.h, [1699](#), [1700](#)
- AAX_ALIGNMENT_HINT
 - AAX_MiscUtils.h, [1716](#)
- AAX_ASSERT
 - AAX_Assert.h, [1327](#)
- AAX_Assert.h, [1323](#), [1329](#)
 - AAX_ASSERT, [1327](#)
 - AAX_DEBUGASSERT, [1327](#)
 - AAX_ETracePriority, [1329](#)
 - AAX_STACKTRACE, [1328](#)
 - AAX_STACKTRACE_RELEASE, [1326](#)
 - AAX_TRACE, [1328](#)
 - AAX_TRACE_RELEASE, [1326](#)
 - AAX_TRACEORSTACKTRACE, [1328](#)
 - AAX_TRACEORSTACKTRACE_RELEASE, [1326](#)
 - kAAX_Trace_Priority_Critical, [1325](#)
 - kAAX_Trace_Priority_High, [1325](#)
 - kAAX_Trace_Priority_Low, [1325](#)
 - kAAX_Trace_Priority_Lowest, [1325](#)
 - kAAX_Trace_Priority_None, [1325](#)
 - kAAX_Trace_Priority_Normal, [1325](#)
- AAX_Atomic.h, [1331](#), [1335](#)
 - AAX_Atomic_CompareAndExchange_32, [1333](#)
 - AAX_Atomic_CompareAndExchange_64, [1334](#)
 - AAX_Atomic_CompareAndExchange_Pointer, [1334](#)
 - AAX_Atomic_DecThenGet_32, [1332](#)
 - AAX_Atomic_Exchange_32, [1332](#)
 - AAX_Atomic_Exchange_64, [1332](#)
 - AAX_Atomic_Exchange_Pointer, [1333](#)
 - AAX_ATOMIC_H, [1332](#)
 - AAX_Atomic_IncThenGet_32, [1332](#)
 - AAX_Atomic_Load_Pointer, [1335](#)
- AAX_Atomic_CompareAndExchange_32
 - AAX_Atomic.h, [1333](#)

- AAX_Atomic_CompareAndExchange_64
 - AAX_Atomic.h, [1334](#)
- AAX_Atomic_CompareAndExchange_Pointer
 - AAX_Atomic.h, [1334](#)
- AAX_Atomic_DecThenGet_32
 - AAX_Atomic.h, [1332](#)
- AAX_Atomic_Exchange_32
 - AAX_Atomic.h, [1332](#)
- AAX_Atomic_Exchange_64
 - AAX_Atomic.h, [1332](#)
- AAX_Atomic_Exchange_Pointer
 - AAX_Atomic.h, [1333](#)
- AAX_ATOMIC_H_
 - AAX_Atomic.h, [1332](#)
- AAX_Atomic_IncThenGet_32
 - AAX_Atomic.h, [1332](#)
- AAX_Atomic_Load_Pointer
 - AAX_Atomic.h, [1335](#)
- AAX_AuxInterface_DirectData.doxygen, [1294](#)
- AAX_AuxInterface_HostProcessor.doxygen, [1294](#)
- AAX_AuxInterface_TaskAgent.doxygen, [1294](#)
- AAX_BigEndianNativeSwap
 - AAX_EndianSwap.h, [1432](#)
- AAX_BigEndianNativeSwapInPlace
 - AAX_EndianSwap.h, [1431](#)
- AAX_BigEndianNativeSwapSequenceInPlace
 - AAX_EndianSwap.h, [1434](#)
- AAX_BugList.doxygen, [1294](#)
- AAX_CALLBACK
 - AAX.h, [1310](#)
- AAX_Callbacks.h, [1339](#), [1342](#)
 - AAX_CBackgroundProc, [1341](#)
 - AAX_CInitPrivateDataProc, [1341](#)
 - AAX_CInstanceInitProc, [1340](#)
 - AAX_CPacketAllocator, [1340](#)
 - AAX_CProcessProc, [1339](#)
 - AAX_CProcPtrID, [1342](#)
 - AAXCreateObjectProc, [1339](#)
 - kAAX_ProcPtrID_Create_EffectDirectData, [1342](#)
 - kAAX_ProcPtrID_Create_EffectGUI, [1342](#)
 - kAAX_ProcPtrID_Create_EffectParameters, [1342](#)
 - kAAX_ProcPtrID_Create_HostProcessor, [1342](#)
 - kAAX_ProcPtrID_Create_SessionDocumentClient, [1342](#)
 - kAAX_ProcPtrID_Create_TaskAgent, [1342](#)
- AAX_CAPTURE
 - AAX_Exception.h, [1504](#)
- AAX_CAPTURE_MULT
 - AAX_Exception.h, [1505](#)
- AAX_CArrayDataBuffer
 - AAX_CArrayDataBuffer< D >, [418](#)
- AAX_CArrayDataBuffer< D >, [416](#)
 - ~AAX_CArrayDataBuffer, [419](#)
 - AAX_CArrayDataBuffer, [418](#)
 - Data, [420](#)
 - operator=, [419](#)
 - Size, [419](#)
 - Type, [419](#)
- AAX_CArrayDataBuffer.h, [1343](#), [1344](#)
 - AAX_CArrayDataBuffer_H, [1344](#)
- AAX_CArrayDataBuffer_H
 - AAX_CArrayDataBuffer.h, [1344](#)
- AAX_CArrayDataBufferOfType
 - AAX_CArrayDataBufferOfType< T, D >, [422](#)
- AAX_CArrayDataBufferOfType< T, D >, [420](#)
 - ~AAX_CArrayDataBufferOfType, [423](#)
 - AAX_CArrayDataBufferOfType, [422](#)
 - Data, [424](#)
 - operator=, [423](#)
 - Size, [423](#)
 - Type, [423](#)
- AAX_CAtomicQueue
 - AAX_CAtomicQueue< T, S >, [427](#)
- AAX_CAtomicQueue< T, S >, [424](#)
 - ~AAX_CAtomicQueue, [427](#)
 - AAX_CAtomicQueue, [427](#)
 - Clear, [427](#)
 - Peek, [428](#)
 - Pop, [428](#)
 - Push, [427](#)
 - template_size, [429](#)
 - template_type, [426](#)
 - value_type, [427](#)
- AAX_CAtomicQueue.h, [1346](#)
- AAX_CAudioInPort
 - AAX.h, [1316](#)
- AAX_CAudioOutPort
 - AAX.h, [1316](#)
- AAX_CAutoreleasePool, [429](#)
 - ~AAX_CAutoreleasePool, [430](#)
 - AAX_CAutoreleasePool, [429](#)
- AAX_CAutoreleasePool.h, [1350](#)
 - _AAX_CAUTORELEASEPOOL_H_, [1350](#)
- AAX_CBackgroundProc
 - AAX_Callbacks.h, [1341](#)
- AAX_CBinaryDisplayDelegate
 - AAX_CBinaryDisplayDelegate< T >, [432](#)
- AAX_CBinaryDisplayDelegate< T >, [430](#)
 - AAX_CBinaryDisplayDelegate, [432](#)
 - AddShortenedStrings, [434](#)
 - Clone, [432](#)
 - StringToValue, [434](#)
 - ValueToString, [433](#)
- AAX_CBinaryDisplayDelegate.h, [1351](#)
- AAX_CBinaryTaperDelegate
 - AAX_CBinaryTaperDelegate< T >, [436](#)
- AAX_CBinaryTaperDelegate< T >, [435](#)
 - AAX_CBinaryTaperDelegate, [436](#)
 - Clone, [437](#)
 - ConstrainRealValue, [437](#)
 - GetMaximumValue, [437](#)
 - GetMinimumValue, [437](#)
 - NormalizedToReal, [438](#)
 - RealToNormalized, [438](#)
- AAX_CBinaryTaperDelegate.h, [1354](#)
- AAX_CBoolean

- AAX.h, 1312
- AAX_CChunkDataParser, 439
 - ~AAX_CChunkDataParser, 441
 - AAX_CChunkDataParser, 441
 - AddDouble, 442
 - AddFloat, 442
 - AddInt16, 442
 - AddInt32, 442
 - AddString, 442
 - Clear, 444
 - FindDouble, 443
 - FindFloat, 443
 - FindInt16, 443
 - FindInt32, 443
 - FindName, 445
 - FindString, 443
 - GetChunkData, 444
 - GetChunkDataSize, 444
 - GetChunkVersion, 444
 - IsEmpty, 444
 - LoadChunk, 445
 - mChunkData, 446
 - mChunkVersion, 446
 - mDataValues, 446
 - mLastFoundIndex, 445
 - ReplaceDouble, 444
 - WordAlign, 445
- AAX_CChunkDataParser.h, 1355, 1357
 - AAX_CHUNKDATAPARSER_H, 1356
- AAX_CChunkDataParser::DataValue, 1277
 - DataValue, 1277
 - mDataName, 1278
 - mDataType, 1278
 - mIntValue, 1278
 - mStringValue, 1278
- AAX_CComponentID
 - AAX.h, 1314
- AAX_CCount
 - AAX.h, 1312
- AAX_CDecibelDisplayDelegateDecorator
 - AAX_CDecibelDisplayDelegateDecorator< T >, 449
- AAX_CDecibelDisplayDelegateDecorator< T >, 446
 - AAX_CDecibelDisplayDelegateDecorator, 449
 - Clone, 449
 - StringToValue, 451
 - ValueToString, 449, 450
- AAX_CDecibelDisplayDelegateDecorator.h, 1358
- AAX_CEffectDirectData, 452
 - ~AAX_CEffectDirectData, 455
 - AAX_CEffectDirectData, 455
 - Controller, 457
 - EffectParameters, 457
 - Initialize, 455
 - Initialize_PrivateDataAccess, 457
 - NotificationReceived, 456
 - TimerWakeup, 456
 - TimerWakeup_PrivateDataAccess, 457
 - Uninitialize, 455
- AAX_CEffectDirectData.h, 1360, 1361
 - AAX_CEFFECTDIRECTDATA_H, 1361
- AAX_CEFFECTDIRECTDATA_H
 - AAX_CEffectDirectData.h, 1361
- AAX_CEffectGUI, 458
 - ~AAX_CEffectGUI, 462
 - AAX_CEffectGUI, 462
 - CreateViewContainer, 466
 - CreateViewContents, 466
 - DeleteViewContainer, 466
 - Draw, 464
 - GetController, 467
 - GetCustomLabel, 465
 - GetEffectParameters, 467
 - GetViewContainer, 468
 - GetViewContainerPtr, 468
 - GetViewContainerType, 468
 - GetViewSize, 464
 - Initialize, 462
 - NotificationReceived, 462
 - ParameterUpdated, 465
 - SetControlHighlightInfo, 466
 - SetViewContainer, 463
 - TimerWakeup, 464
 - Transport, 468
 - Uninitialize, 462
 - UpdateAllParameters, 467
- AAX_CEffectGUI.h, 1362
- AAX_CEffectID
 - AAX.h, 1315
- AAX_CEffectParameters, 469
 - ~AAX_CEffectParameters, 478
 - AAX_CEffectParameters, 478
 - AutomationDelegate, 503
 - BuildChunkData, 505
 - CompareActiveChunk, 495
 - Controller, 502
 - DoMIDITransfers, 500
 - EffectInit, 504
 - FilterParameterIDOnSave, 504
 - GenerateCoefficients, 491
 - GetChunk, 494
 - GetChunkIDFromIndex, 493
 - GetChunkSize, 493
 - GetCurveData, 496
 - GetCurveDataDisplayRange, 498
 - GetCurveDataMeterIds, 497
 - GetCustomData, 499
 - GetMasterBypassParameter, 480
 - GetNumberOfChanges, 495
 - GetNumberOfChunks, 493
 - GetNumberOfParameters, 480
 - GetParameter, 484
 - GetParameterDefaultNormalizedValue, 482
 - GetParameterIDFromIndex, 485
 - GetParameterIndex, 484
 - GetParameterIsAutomatable, 480

- GetParameterName, 481
- GetParameterNameOfLength, 481
- GetParameterNormalizedValue, 487
- GetParameterNumberOfSteps, 481
- GetParameterOrientation, 483
- GetParameterStringFromValue, 486
- GetParameterType, 483
- GetParameterValueFromString, 485
- GetParameterValueInfo, 485
- GetParameterValueString, 486
- Initialize, 478
- IsParameterLinkReady, 503
- IsParameterTouched, 503
- mChunkParser, 505
- mChunkSize, 505
- mFilteredParameters, 506
- mNumChunkedParameters, 505
- mNumPluginChanges, 505
- mPacketDispatcher, 506
- mParameterManager, 506
- NotificationReceived, 479
- operator=, 478
- ReleaseParameter, 489
- RenderAudio_Hybrid, 501
- ResetFieldData, 492
- SetChunk, 495
- SetCustomData, 500
- SetDisplayDelegate, 503
- SetParameterDefaultNormalizedValue, 482
- SetParameterNormalizedRelative, 488
- SetParameterNormalizedValue, 487
- SetTaperDelegate, 503
- TimerWakeup, 496
- TouchParameter, 488
- Transport, 502
- Uninitialize, 479
- UpdateControlMIDINodes, 501
- UpdateMIDINodes, 500
- UpdatePageTable, 499, 504
- UpdateParameterNormalizedRelative, 490
- UpdateParameterNormalizedValue, 490
- UpdateParameterTouch, 489
- AAX_CEffectParameters.h, 1363, 1365
 - BoolToNormalized, 1364
 - cDefaultMasterBypassID, 1365
 - cPreviewID, 1365
 - Int32ToNormalized, 1364
 - NormalizedToInt32, 1364
- AAX_CFieldIndex
 - AAX.h, 1314
- AAX_CheckedResult, 506
 - ~AAX_CheckedResult, 509
 - AAX_CheckedResult, 509
 - AddAcceptedResult, 509
 - Clear, 511
 - Exception, 508
 - LastError, 511
 - operator AAX_Result, 510
 - operator=, 509
 - operator |=, 510
 - ResetAcceptedResults, 509
- AAX_CHostProcessor, 511
 - ~AAX_CHostProcessor, 516
 - AAX_CHostProcessor, 516
 - AnalyzeAudio, 519
 - Controller, 524
 - EffectParameters, 525
 - GetAudio, 524
 - GetClipNameSuffix, 521
 - GetDstEnd, 523
 - GetDstStart, 523
 - GetEffectParameters, 521
 - GetHostProcessorDelegate, 521, 522
 - GetInputRange, 522
 - GetLocation, 522
 - GetOutputRange, 522
 - GetSideChainInputNum, 524
 - GetSrcEnd, 522
 - GetSrcStart, 522
 - HostProcessorDelegate, 524, 525
 - Initialize, 516
 - InitOutputBounds, 517
 - PostAnalyze, 520
 - PostRender, 519
 - PreAnalyze, 520
 - PreRender, 519
 - RenderAudio, 518
 - SetLocation, 518
 - TranslateOutputBounds, 523
 - Uninitialize, 516
- AAX_CHostProcessor.h, 1367, 1368
- AAX_CHostServices, 525
 - HandleAssertFailure, 526
 - Set, 526
 - StackTrace, 527
 - Trace, 526
- AAX_CHostServices.h, 1369
- AAX_CHUNKDATAPARSER_H
 - AAX_CChunkDataParser.h, 1356
- AAX_ChunkDataParserDefs, 407
 - BUILD_DATA_FAILED, 410
 - DEFAULT32BIT_TYPE_INCR, 410
 - DEFAULT32BIT_TYPE_SIZE, 409
 - DOUBLE_STRING_IDENTIFIER, 408
 - DOUBLE_TYPE, 408
 - DOUBLE_TYPE_INCR, 408
 - DOUBLE_TYPE_SIZE, 408
 - FLOAT_STRING_IDENTIFIER, 408
 - FLOAT_TYPE, 408
 - HEADER_SIZE, 410
 - LONG_STRING_IDENTIFIER, 408
 - LONG_TYPE, 408
 - MAX_NAME_LENGTH, 410
 - MAX_STRINGDATA_LENGTH, 409
 - NAME_NOT_FOUND, 410
 - SHORT_STRING_IDENTIFIER, 409

- SHORT_TYPE, [409](#)
- SHORT_TYPE_INCR, [409](#)
- SHORT_TYPE_SIZE, [409](#)
- STRING_IDENTIFIER_SIZE, [410](#)
- STRING_STRING_IDENTIFIER, [409](#)
- STRING_TYPE, [409](#)
- VERSION_ID_1, [410](#)
- AAX_CIndex
 - AAX.h, [1311](#)
- AAX_CInitPrivateDataProc
 - AAX_Callbacks.h, [1341](#)
- AAX_CInstanceInitProc
 - AAX_Callbacks.h, [1340](#)
- AAX_CLinearTaperDelegate
 - AAX_CLinearTaperDelegate< T, RealPrecision >, [530](#)
- AAX_CLinearTaperDelegate< T, RealPrecision >, [527](#)
 - AAX_CLinearTaperDelegate, [530](#)
 - Clone, [530](#)
 - ConstrainRealValue, [531](#)
 - GetMaximumValue, [531](#)
 - GetMinimumValue, [530](#)
 - NormalizedToReal, [531](#)
 - RealToNormalized, [532](#)
 - Round, [532](#)
- AAX_CLinearTaperDelegate.h, [1370](#)
- AAX_CLogTaperDelegate
 - AAX_CLogTaperDelegate< T, RealPrecision >, [535](#)
- AAX_CLogTaperDelegate< T, RealPrecision >, [533](#)
 - AAX_CLogTaperDelegate, [535](#)
 - Clone, [535](#)
 - ConstrainRealValue, [536](#)
 - GetMaximumValue, [536](#)
 - GetMinimumValue, [535](#)
 - NormalizedToReal, [536](#)
 - RealToNormalized, [537](#)
 - Round, [538](#)
- AAX_CLogTaperDelegate.h, [1372](#)
- AAX_CMeterID
 - AAX.h, [1314](#)
- AAX_CMeterPort
 - AAX.h, [1316](#)
- AAX_CMidiPacket, [538](#)
 - mData, [539](#)
 - mIsImmediate, [539](#)
 - mLength, [539](#)
 - mTimestamp, [539](#)
- AAX_CMidiStream, [540](#)
 - mBuffer, [541](#)
 - mBufferSize, [540](#)
- AAX_CMonolithicParameters, [541](#)
 - ~AAX_CMonolithicParameters, [550](#)
 - AAX_CMonolithicParameters, [550](#)
 - AddSynchronizedParameter, [551](#)
 - GenerateCoefficients, [553](#)
 - RenderAudio, [550](#)
 - ResetFieldData, [554](#)
 - StaticDescribe, [555](#)
 - StaticRenderAudio, [557](#)
 - TimerWakeup, [555](#)
 - TParamValPair, [549](#)
 - UpdateParameterNormalizedValue, [552](#)
- AAX_CMonolithicParameters.cpp, [1298](#)
- AAX_CMonolithicParameters.h, [1298](#), [1300](#)
 - kMaxAdditionalMIDINodes, [1299](#)
 - kMaxAuxOutputStems, [1299](#)
 - kSynchronizedParameterQueueSize, [1299](#)
- AAX_CMutex, [558](#)
 - ~AAX_CMutex, [559](#)
 - AAX_CMutex, [559](#)
 - Lock, [559](#)
 - Try_Lock, [560](#)
 - Unlock, [559](#)
- AAX_CMutex.h, [1373](#), [1374](#)
- AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >, [560](#)
 - Clone, [562](#)
 - StringToValue, [564](#)
 - ValueToString, [562](#), [563](#)
- AAX_CNumberDisplayDelegate.h, [1374](#), [1375](#)
- AAX_CommonConversions.h, [1376](#), [1383](#)
 - DBToGain, [1377](#)
 - DoubleTo32BitDSPCoef, [1380](#)
 - DoubleTo32BitDSPCoefRnd, [1379](#)
 - DoubleToDSPCoef, [1378](#)
 - DoubleToDSPCoefRnd, [1380](#)
 - DoubleToLong, [1378](#)
 - DSPCoefToDouble, [1378](#)
 - GainToDB, [1377](#)
 - k32BitAbsMax, [1380](#)
 - k32BitNegMax, [1381](#)
 - k32BitPosMax, [1380](#)
 - k56kFloatNegMax, [1382](#)
 - k56kFloatPosMax, [1382](#)
 - k56kFracAbsMax, [1381](#)
 - k56kFracHalf, [1381](#)
 - k56kFracNegMax, [1381](#)
 - k56kFracNegOne, [1381](#)
 - k56kFracPosMax, [1381](#)
 - k56kFracZero, [1381](#)
 - kNeg144DB, [1382](#)
 - kNeg144Gain, [1382](#)
 - kOneOver56kFracAbsMax, [1382](#)
 - LongToDouble, [1377](#)
 - ThirtyTwoBitDSPCoefToDouble, [1379](#)
- AAX_CommonInterface_Algorithm.doxygen, [1294](#)
- AAX_CommonInterface_Communication.doxygen, [1294](#)
- AAX_CommonInterface_DataModel.doxygen, [1294](#)
- AAX_CommonInterface_Describe.doxygen, [1294](#)
- AAX_CommonInterface_FormatSpecification.doxygen, [1295](#)
- AAX_CommonInterface_GUI.doxygen, [1295](#)
- AAX_ComplID_DescriptionHost
 - AAX_UIDs.h, [1662](#)
- AAX_ComplID_FeatureInfo

- AAX_UIDs.h, 1663
- AAX_Component< aContextType >, 565
 - CBackgroundProc, 566
 - CInitPrivateDataProc, 566
 - CInstanceInitProc, 566
 - CPacketAllocator, 566
 - CProcessProc, 565
- AAX_Constants.h, 1700, 1702
 - AAX_CONSTANTS_H, 1701
- AAX_CONSTANTS_H
 - AAX_Constants.h, 1701
- AAX_CONSTEXPR
 - AAX.h, 1308
- AAX_CPacket, 566
 - ~AAX_CPacket, 567
 - AAX_CPacket, 567
 - GetID, 568
 - GetPtr, 567, 568
 - GetSize, 568
 - IsDirty, 568
 - SetDirty, 567
- AAX_CPacketAllocator
 - AAX_Callbacks.h, 1340
- AAX_CPacketDispatcher, 568
 - ~AAX_CPacketDispatcher, 569
 - AAX_CPacketDispatcher, 569
 - Dispatch, 571
 - GenerateSingleValuePacket, 571
 - Initialize, 569
 - RegisterPacket, 569, 570
 - SetDirty, 571
- AAX_CPacketDispatcher.h, 1384, 1385
- AAX_CPacketHandler
 - AAX_CPacketHandler< TWorker >, 573
- AAX_CPacketHandler< TWorker >, 572
 - AAX_CPacketHandler, 573
 - Call, 574
 - Clone, 574
 - fpt, 574
 - fptEx, 574
 - pt2Object, 574
- AAX_CPageTableParamID
 - AAX.h, 1315
- AAX_CParameter
 - AAX_CParameter< T >, 581–583
- AAX_CParameter< T >, 575
 - ~AAX_CParameter, 584
 - AAX_CParameter, 581–583
 - AAX_DEFAULT_MOVE_CTOR, 584
 - AAX_DEFAULT_MOVE_OPER, 584
 - AAX_DELETE, 584, 585
 - AddShortenedName, 586
 - Automatable, 600
 - ClearShortenedNames, 587
 - CloneValue, 585
 - Defaults, 580
 - DisplayDelegate, 607
 - eParameterDefaultNumStepsContinuous, 581
 - eParameterDefaultNumStepsDiscrete, 581
 - eParameterTypeBool, 580
 - eParameterTypeCustom, 580
 - eParameterTypeFloat, 580
 - eParameterTypeInt32, 580
 - eParameterTypeUndefined, 580
 - GetBoolFromNormalizedValue, 596, 611
 - GetDefaultValue, 606
 - GetDoubleFromNormalizedValue, 597, 612
 - GetFloatFromNormalizedValue, 597, 612
 - GetInt32FromNormalizedValue, 596, 612
 - GetNormalizedDefaultValue, 587
 - GetNormalizedValue, 588
 - GetNormalizedValueFromBool, 594, 609
 - GetNormalizedValueFromDouble, 595, 611
 - GetNormalizedValueFromFloat, 595, 610
 - GetNormalizedValueFromInt32, 594, 610
 - GetNormalizedValueFromStep, 589
 - GetNormalizedValueFromString, 596
 - GetNumberOfSteps, 589
 - GetOrientation, 591
 - GetStepValue, 589
 - GetStepValueFromNormalizedValue, 590
 - GetStringFromNormalizedValue, 598
 - GetType, 591
 - GetValue, 606
 - GetValueAsBool, 601
 - GetValueAsDouble, 602
 - GetValueAsFloat, 602
 - GetValueAsInt32, 601
 - GetValueAsString, 602, 607
 - GetValueString, 593
 - Identifier, 585
 - mAutomatable, 613
 - mAutomationDelegate, 614
 - mControlType, 613
 - mDefaultValue, 614
 - mDisplayDelegate, 614
 - mNames, 613
 - mNeedNotify, 614
 - mNumSteps, 613
 - mOrientation, 614
 - mTaperDelegate, 614
 - mValue, 614
 - Name, 586
 - Release, 600
 - SetAutomationDelegate, 599
 - SetDefaultValue, 606
 - SetDisplayDelegate, 592
 - SetName, 585
 - SetNormalizedDefaultValue, 587
 - SetNormalizedValue, 588
 - SetNumberOfSteps, 588
 - SetOrientation, 591
 - SetStepValue, 590
 - SetTaperDelegate, 591
 - SetToDefaultValue, 587
 - SetType, 590

- SetValue, 605
- SetValueFromString, 599
- SetValueWithBool, 603, 607
- SetValueWithDouble, 604, 608
- SetValueWithFloat, 604, 608
- SetValueWithInt32, 603, 608
- SetValueWithString, 605, 609
- ShortenedName, 586
- TaperDelegate, 606
- Touch, 600
- Type, 580
- UpdateNormalizedValue, 605
- AAX_CParameter.h, 1387
- AAX_CParameterManager, 615
 - ~AAX_CParameterManager, 616
 - AAX_CParameterManager, 616
 - AddParameter, 620
 - GetParameter, 619
 - GetParameterByID, 617, 618
 - GetParameterByName, 618
 - GetParameterIndex, 619
 - Initialize, 616
 - mAutomationDelegate, 620
 - mParameters, 621
 - mParametersMap, 621
 - NumParameters, 617
 - RemoveAllParameters, 617
 - RemoveParameter, 620
 - RemoveParameterByID, 617
- AAX_CParameterManager.h, 1400, 1401
- AAX_CParameterValue
 - AAX_CParameterValue< T >, 623, 624
- AAX_CParameterValue< T >, 621
 - AAX_CParameterValue, 623, 624
 - AAX_DEFAULT_DTOR_OVERRIDE, 624
 - AAX_DEFAULT_MOVE_CTOR, 624
 - AAX_DEFAULT_MOVE_OPER, 625
 - AAX_DELETE, 625
 - Clone, 625
 - Defaults, 623
 - eParameterDefaultMaxIdentifierLength, 623
 - eParameterDefaultMaxIdentifierSize, 623
 - Get, 625
 - GetValueAsBool, 626, 628
 - GetValueAsDouble, 627, 629
 - GetValueAsFloat, 627, 629
 - GetValueAsInt32, 626, 629
 - GetValueAsString, 628, 630
 - Identifier, 626
 - Set, 625
- AAX_CParamID
 - AAX.h, 1314
- AAX_CPercentDisplayDelegateDecorator
 - AAX_CPercentDisplayDelegateDecorator< T >, 633
- AAX_CPercentDisplayDelegateDecorator< T >, 630
 - AAX_CPercentDisplayDelegateDecorator, 633
 - Clone, 633
 - StringToValue, 635
 - ValueToString, 633, 634
- AAX_CPercentDisplayDelegateDecorator.h, 1401, 1402
 - AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H, 1402
- AAX_CPERCENTDISPLAYDELEGATEDECORATOR_H
 - AAX_CPercentDisplayDelegateDecorator.h, 1402
- AAX_CPieceWiseLinearTaperDelegate
 - AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, 638
- AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, 636
 - ~AAX_CPieceWiseLinearTaperDelegate, 638
 - AAX_CPieceWiseLinearTaperDelegate, 638
 - Clone, 639
 - ConstrainRealValue, 639
 - GetMaximumValue, 639
 - GetMinimumValue, 639
 - NormalizedToReal, 640
 - RealToNormalized, 640
 - Round, 640
- AAX_CPieceWiseLinearTaperDelegate.h, 1403, 1404
- AAX_CPointerPropertyValue
 - AAX.h, 1313
- AAX_CPP11_SUPPORT
 - AAX.h, 1306
- AAX_CProcessProc
 - AAX_Callbacks.h, 1339
- AAX_CProcPtrID
 - AAX_Callbacks.h, 1342
- AAX_CPropertyValue
 - AAX.h, 1313
- AAX_CPropertyValue64
 - AAX.h, 1313
- AAX_CRangeTaperDelegate
 - AAX_CRangeTaperDelegate< T, RealPrecision >, 643, 644
- AAX_CRangeTaperDelegate< T, RealPrecision >, 641
 - AAX_CRangeTaperDelegate, 643, 644
 - Clone, 644
 - ConstrainRealValue, 645
 - GetMaximumValue, 645
 - GetMinimumValue, 645
 - NormalizedToReal, 646
 - operator=, 644
 - RealToNormalized, 646
 - Round, 646
 - SmartRound, 647
- AAX_CRangeTaperDelegate.h, 1407
- AAX_CSampleRate
 - AAX.h, 1313
- AAX_CSelector
 - AAX.h, 1312
- AAX_CSessionDocumentClient, 647
 - ~AAX_CSessionDocumentClient, 650
 - AAX_CSessionDocumentClient, 650
 - GetController, 652
 - GetEffectParameters, 652

- GetSessionDocument, 653
- Initialize, 650
- NotificationReceived, 651
- SessionDocumentChanged, 652
- SessionDocumentWillChange, 651
- SetSessionDocument, 650
- Uninitialize, 650
- AAX_CSessionDocumentClient.h, 1410
 - AAX_CSessionDocumentClient_H, 1410
- AAX_CSessionDocumentClient_H
 - AAX_CSessionDocumentClient.h, 1410
- AAX_CStateDisplayDelegate
 - AAX_CStateDisplayDelegate< T >, 655, 656
- AAX_CStateDisplayDelegate< T >, 653
 - AAX_CStateDisplayDelegate, 655, 656
 - AddShortenedStrings, 658
 - Clone, 656
 - Compare, 658
 - StringToValue, 657
 - ValueToString, 656, 657
- AAX_CStateDisplayDelegate.h, 1411, 1412
- AAX_CStatelessParameter, 658
 - AAX_CStatelessParameter, 662
 - AAX_DEFAULT_DTOR_OVERRIDE, 662
 - AddShortenedName, 664
 - Automatable, 666
 - ClearShortenedNames, 665
 - CloneValue, 662
 - GetBoolFromNormalizedValue, 674
 - GetDoubleFromNormalizedValue, 676
 - GetFloatFromNormalizedValue, 675
 - GetInt32FromNormalizedValue, 675
 - GetNormalizedDefaultValue, 669
 - GetNormalizedValue, 668
 - GetNormalizedValueFromBool, 672
 - GetNormalizedValueFromDouble, 674
 - GetNormalizedValueFromFloat, 673
 - GetNormalizedValueFromInt32, 673
 - GetNormalizedValueFromStep, 670
 - GetNormalizedValueFromString, 674
 - GetNumberOfSteps, 670
 - GetOrientation, 683
 - GetStepValue, 670
 - GetStepValueFromNormalizedValue, 670
 - GetStringFromNormalizedValue, 676, 677
 - GetType, 683
 - GetValueAsBool, 678
 - GetValueAsDouble, 679
 - GetValueAsFloat, 679
 - GetValueAsInt32, 679
 - GetValueAsString, 680
 - GetValueString, 671
 - Identifier, 662
 - mAutomationDelegate, 685
 - mID, 685
 - mNames, 685
 - mValueString, 685
 - Name, 664
 - Release, 667
 - SetAutomationDelegate, 666
 - SetDisplayDelegate, 684
 - SetName, 663
 - SetNormalizedDefaultValue, 668
 - SetNormalizedValue, 668
 - SetNumberOfSteps, 669
 - SetOrientation, 683
 - SetStepValue, 670
 - SetTaperDelegate, 684
 - SetToDefaultValue, 669
 - SetType, 682
 - SetValueFromString, 678
 - SetValueWithBool, 680
 - SetValueWithDouble, 682
 - SetValueWithFloat, 681
 - SetValueWithInt32, 681
 - SetValueWithString, 682
 - ShortenedName, 665
 - Touch, 667
 - UpdateNormalizedValue, 684
- AAX_CStateTaperDelegate
 - AAX_CStateTaperDelegate< T >, 687
- AAX_CStateTaperDelegate< T >, 686
 - AAX_CStateTaperDelegate, 687
 - Clone, 688
 - ConstrainRealValue, 688
 - GetMaximumValue, 688
 - GetMinimumValue, 688
 - NormalizedToReal, 689
 - RealToNormalized, 689
- AAX_CStateTaperDelegate.h, 1414, 1415
- AAX_CString, 690
 - AAX_CString, 692, 693
 - AAX_DEFAULT_MOVE_CTOR, 695
 - Append, 697
 - AppendHex, 698
 - AppendNumber, 698
 - Clear, 696
 - CString, 700
 - Empty, 696
 - Equals, 702, 703
 - Erase, 697
 - FindFirst, 700
 - FindLast, 700
 - Get, 694
 - Insert, 698, 699
 - InsertHex, 699
 - InsertNumber, 699
 - kInvalidIndex, 706
 - kMaxStringLength, 706
 - Length, 693
 - MaxLength, 694
 - mString, 706
 - operator!=, 704
 - operator<, 704
 - operator<<, 706
 - operator>, 705

- operator>>, 706
- operator+&, 705
- operator=, 695, 696
- operator==, 703, 704
- operator[], 705
- Replace, 699
- Set, 695
- StdString, 695, 696
- SubString, 701
- ToDouble, 701
- ToInteger, 701
- AAX_CString.h, 1416, 1418
 - AAX_CSTRING_H, 1417
 - operator+, 1417
- AAX_CSTRING_H
 - AAX_CString.h, 1417
- AAX_CStringAbbreviations, 707
 - AAX_CStringAbbreviations, 707
 - Add, 708
 - Clear, 709
 - Get, 708
 - Primary, 707
 - SetPrimary, 707
- AAX_CStringDataBuffer, 710
 - ~AAX_CStringDataBuffer, 712
 - AAX_CStringDataBuffer, 711, 712
 - Data, 713
 - operator=, 712, 713
 - Size, 713
 - Type, 713
- AAX_CStringDataBuffer.h, 1420, 1421
 - AAX_CStringDataBuffer_H, 1420
- AAX_CStringDataBuffer_H
 - AAX_CStringDataBuffer.h, 1420
- AAX_CStringDataBufferOfType
 - AAX_CStringDataBufferOfType< T >, 716
- AAX_CStringDataBufferOfType< T >, 714
 - ~AAX_CStringDataBufferOfType, 716
 - AAX_CStringDataBufferOfType, 716
 - Data, 717
 - operator=, 717
 - Size, 717
 - Type, 717
- AAX_CStringDisplayDelegate
 - AAX_CStringDisplayDelegate< T >, 720
- AAX_CStringDisplayDelegate< T >, 718
 - AAX_CStringDisplayDelegate, 720
 - Clone, 720
 - mInverseStringMap, 722
 - mStringMap, 722
 - StringToValue, 722
 - ValueToString, 721
- AAX_CStringDisplayDelegate.h, 1422
- AAX_CTargetPlatform
 - AAX.h, 1314
- AAX_CTaskAgent, 723
 - ~AAX_CTaskAgent, 725
 - AAX_CTaskAgent, 725
 - AddTask, 726, 727
 - CancelAllTasks, 726
 - GetController, 727
 - GetEffectParameters, 727
 - Initialize, 726
 - ReceiveTask, 727
 - Uninitialize, 726
- AAX_CTaskAgent.h, 1424
- AAX_CTempoBreakpoint, 728
 - mSampleLocation, 728
 - mValue, 728
- AAX_CTimeOfDay
 - AAX.h, 1312
- AAX_CTimestamp
 - AAX.h, 1312
- AAX_CTransportCounter
 - AAX.h, 1312
- AAX_CTypeID
 - AAX.h, 1313
- AAX_CUnitDisplayDelegateDecorator
 - AAX_CUnitDisplayDelegateDecorator< T >, 731
- AAX_CUnitDisplayDelegateDecorator< T >, 728
 - AAX_CUnitDisplayDelegateDecorator, 731
 - Clone, 731
 - mUnitString, 734
 - StringToValue, 733
 - ValueToString, 731, 732
- AAX_CUnitDisplayDelegateDecorator.h, 1425
- AAX_CUnitPrefixDisplayDelegateDecorator
 - AAX_CUnitPrefixDisplayDelegateDecorator< T >, 737
- AAX_CUnitPrefixDisplayDelegateDecorator< T >, 734
 - AAX_CUnitPrefixDisplayDelegateDecorator, 737
 - Clone, 737
 - StringToValue, 739
 - ValueToString, 738
- AAX_CUnitPrefixDisplayDelegateDecorator.h, 1427
- AAX_DEBUGASSERT
 - AAX_Assert.h, 1327
- AAX_DEFAULT_ASGN_OPER
 - AAX.h, 1308
- AAX_DEFAULT_COPY_CTOR
 - AAX.h, 1307
- AAX_DEFAULT_CTOR
 - AAX.h, 1307
- AAX_DEFAULT_DTOR
 - AAX.h, 1307
- AAX_DEFAULT_DTOR_OVERRIDE
 - AAX.h, 1307
 - AAX_CParameterValue< T >, 624
 - AAX_CStatelessParameter, 662
- AAX_DEFAULT_MOVE_CTOR
 - AAX.h, 1307
 - AAX::Exception::Any, 1275
 - AAX_CParameter< T >, 584
 - AAX_CParameterValue< T >, 624
 - AAX_CString, 695
- AAX_DEFAULT_MOVE_OPER

- AAX.h, [1308](#)
- AAX::Exception::Any, [1275](#)
- AAX_CParameter< T >, [584](#)
- AAX_CParameterValue< T >, [625](#)
- AAX_DELETE
 - AAX.h, [1308](#)
 - AAX_CParameter< T >, [584](#), [585](#)
 - AAX_CParameterValue< T >, [625](#)
 - AAX_IDataBuffer, [987](#)
 - AAX_IEffectDirectData, [1023](#)
 - AAX_IEffectGUI, [1026](#)
 - AAX_IEffectParameters, [1033](#)
 - AAX_IHostProcessor, [1037](#)
 - AAX_ISessionDocumentClient, [1104](#)
 - AAX_ITaskAgent, [1118](#)
- AAX_Denormal.h, [1703](#), [1704](#)
 - AAX_DENORMAL_H, [1703](#)
 - AAX_SCOPE_COMPUTE_DENORMALS, [1704](#)
 - AAX_SCOPE_DENORMALS_AS_ZERO, [1704](#)
- AAX_DENORMAL_H
 - AAX_Denormal.h, [1703](#)
- AAX_DigiTrace_Guide.doxygen, [1295](#)
- AAX_DistributingYourPlugIn.doxygen, [1295](#)
- AAX_DMA_API
 - AAX_IDma.h, [1565](#)
- AAX_DocsDirectory.doxygen, [1295](#)
- AAX_DocumentData_UID
 - AAX_UIDs.h, [1654](#)
- AAX_DocumentDataType_TempoMap
 - AAX_UIDs.h, [1668](#)
- AAX_DWORD_ALIGNED_HINT
 - AAX_MiscUtils.h, [1716](#)
- AAX_EAssertFlags
 - AAX_Enums.h, [1474](#)
- AAX_eAssertFlags_Default
 - AAX_Enums.h, [1474](#)
- AAX_eAssertFlags_Dialog
 - AAX_Enums.h, [1474](#)
- AAX_eAssertFlags_Log
 - AAX_Enums.h, [1474](#)
- AAX_EAudioBufferLength
 - AAX_Enums.h, [1449](#)
- AAX_eAudioBufferLength_1
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_1024
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_128
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_16
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_2
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_256
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_32
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_4
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_512
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_64
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_8
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_Max
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLength_Undefined
 - AAX_Enums.h, [1450](#)
- AAX_EAudioBufferLengthDSP
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLengthDSP_16
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLengthDSP_32
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLengthDSP_4
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLengthDSP_64
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLengthDSP_Default
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLengthDSP_Max
 - AAX_Enums.h, [1450](#)
- AAX_eAudioBufferLengthNative_Max
 - AAX_Enums.h, [1451](#)
- AAX_eAudioBufferLengthNative_Min
 - AAX_Enums.h, [1451](#)
- AAX_EComponentInstanceInitAction
 - AAX_Enums.h, [1463](#)
- AAX_eComponentInstanceInitAction_AddingNewInstance
 - AAX_Enums.h, [1464](#)
- AAX_eComponentInstanceInitAction_RemovingInstance
 - AAX_Enums.h, [1464](#)
- AAX_eComponentInstanceInitAction_ResetInstance
 - AAX_Enums.h, [1464](#)
- AAX_EConstraintLocationMask
 - AAX_Enums.h, [1463](#)
- AAX_eConstraintLocationMask_DataModel
 - AAX_Enums.h, [1463](#)
- AAX_eConstraintLocationMask_DLLChipAffinity
 - AAX_Enums.h, [1463](#)
- AAX_eConstraintLocationMask_None
 - AAX_Enums.h, [1463](#)
- AAX_EConstraintTopology
 - AAX_Enums.h, [1463](#)
- AAX_eConstraintTopology_Monolithic
 - AAX_Enums.h, [1463](#)
- AAX_eConstraintTopology_None
 - AAX_Enums.h, [1463](#)
- AAX_ECurveType
 - EQ and Dynamics Curve Displays, [100](#)
- AAX_eCurveType_Dynamics
 - EQ and Dynamics Curve Displays, [101](#)
- AAX_eCurveType_EQ
 - EQ and Dynamics Curve Displays, [101](#)
- AAX_eCurveType_None
 - EQ and Dynamics Curve Displays, [101](#)

- AAX_eCurveType_Reduction
 - EQ and Dynamics Curve Displays, [101](#)
- AAX_EDataInPortType
 - AAX_Enums.h, [1469](#)
- AAX_eDataInPortType_Buffered
 - AAX_Enums.h, [1469](#)
- AAX_eDataInPortType_Incremental
 - AAX_Enums.h, [1469](#)
- AAX_eDataInPortType_Unbuffered
 - AAX_Enums.h, [1469](#)
- AAX_eEQBandType_HighPass
 - AAX_Enums.h, [1466](#)
- AAX_eEQBandType_HighShelf
 - AAX_Enums.h, [1466](#)
- AAX_eEQBandType_LowPass
 - AAX_Enums.h, [1466](#)
- AAX_eEQBandType_LowShelf
 - AAX_Enums.h, [1466](#)
- AAX_eEQBandType_Notch
 - AAX_Enums.h, [1466](#)
- AAX_eEQBandType_Parametric
 - AAX_Enums.h, [1466](#)
- AAX_EEQBandTypes
 - AAX_Enums.h, [1466](#)
- AAX_EEQInCircuitPolarity
 - AAX_Enums.h, [1467](#)
- AAX_eEQInCircuitPolarity_Bypassed
 - AAX_Enums.h, [1467](#)
- AAX_eEQInCircuitPolarity_Disabled
 - AAX_Enums.h, [1467](#)
- AAX_eEQInCircuitPolarity_Enabled
 - AAX_Enums.h, [1467](#)
- AAX_EError
 - AAX_Errors.h, [1491](#)
- AAX_EFeetFramesRate
 - AAX_Enums.h, [1470](#)
- AAX_eFeetFramesRate_23976
 - AAX_Enums.h, [1471](#)
- AAX_eFeetFramesRate_24
 - AAX_Enums.h, [1471](#)
- AAX_eFeetFramesRate_25
 - AAX_Enums.h, [1471](#)
- AAX_EFrameRate
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_100Frame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_11988DropFrame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_11988NonDrop
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_120DropFrame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_120NonDrop
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_23976
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_24Frame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_25Frame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_2997DropFrame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_2997NonDrop
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_30DropFrame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_30NonDrop
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_47952
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_48Frame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_50Frame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_5994DropFrame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_5994NonDrop
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_60DropFrame
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_60NonDrop
 - AAX_Enums.h, [1470](#)
- AAX_eFrameRate_Undeclared
 - AAX_Enums.h, [1470](#)
- AAX_EHighlightColor
 - AAX_Enums.h, [1448](#)
- AAX_eHighlightColor_Blue
 - AAX_Enums.h, [1448](#)
- AAX_eHighlightColor_Green
 - AAX_Enums.h, [1448](#)
- AAX_eHighlightColor_Num
 - AAX_Enums.h, [1448](#)
- AAX_eHighlightColor_Red
 - AAX_Enums.h, [1448](#)
- AAX_eHighlightColor_Yellow
 - AAX_Enums.h, [1448](#)
- AAX_EHostLevel
 - AAX_Enums.h, [1473](#)
- AAX_eHostLevel_Entry
 - AAX_Enums.h, [1474](#)
- AAX_eHostLevel_Intermediate
 - AAX_Enums.h, [1474](#)
- AAX_eHostLevel_Standard
 - AAX_Enums.h, [1474](#)
- AAX_eHostLevel_Unknown
 - AAX_Enums.h, [1474](#)
- AAX_EHostMode
 - AAX_Enums.h, [1462](#)
- AAX_eHostMode_Config
 - AAX_Enums.h, [1462](#)
- AAX_eHostMode_Show
 - AAX_Enums.h, [1462](#)
- AAX_EHostModeBits
 - AAX_Enums.h, [1461](#)
- AAX_eHostModeBits_Live
 - AAX_Enums.h, [1461](#)

- AAX_eHostModeBits_None
 - AAX_Enums.h, [1461](#)
- AAX_EMaxAudioSuiteTracks
 - AAX_Enums.h, [1451](#)
- AAX_eMaxAudioSuiteTracks
 - AAX_Enums.h, [1451](#)
- AAX_EMeterBallisticType
 - AAX_Enums.h, [1456](#)
- AAX_eMeterBallisticType_Host
 - AAX_Enums.h, [1456](#)
- AAX_eMeterBallisticType_NoDecay
 - AAX_Enums.h, [1456](#)
- AAX_EMeterOrientation
 - AAX_Enums.h, [1455](#)
- AAX_eMeterOrientation_BottomLeft
 - AAX_Enums.h, [1455](#)
- AAX_eMeterOrientation_Center
 - AAX_Enums.h, [1455](#)
- AAX_eMeterOrientation_Default
 - AAX_Enums.h, [1455](#)
- AAX_eMeterOrientation_PhaseDot
 - AAX_Enums.h, [1455](#)
- AAX_eMeterOrientation_TopRight
 - AAX_Enums.h, [1455](#)
- AAX_EMeterType
 - AAX_Enums.h, [1456](#)
- AAX_eMeterType_Analysis
 - AAX_Enums.h, [1456](#)
- AAX_eMeterType_CLGain
 - AAX_Enums.h, [1456](#)
- AAX_eMeterType_EGGain
 - AAX_Enums.h, [1456](#)
- AAX_eMeterType_Input
 - AAX_Enums.h, [1456](#)
- AAX_eMeterType_None
 - AAX_Enums.h, [1456](#)
- AAX_eMeterType_Other
 - AAX_Enums.h, [1456](#)
- AAX_eMeterType_Output
 - AAX_Enums.h, [1456](#)
- AAX_eMIDIBeatClock
 - AAX_Enums.h, [1471](#)
- AAX_eMIDIClick
 - AAX_Enums.h, [1471](#)
- AAX_EMidiGlobalNodeSelectors
 - AAX_Enums.h, [1471](#)
- AAX_eMIDIMtc
 - AAX_Enums.h, [1471](#)
- AAX_EMIDINodeType
 - AAX_Enums.h, [1467](#)
- AAX_eMIDINodeType_Global
 - AAX_Enums.h, [1468](#)
- AAX_eMIDINodeType_LocalInput
 - AAX_Enums.h, [1468](#)
- AAX_eMIDINodeType_LocalOutput
 - AAX_Enums.h, [1468](#)
- AAX_eMIDINodeType_Transport
 - AAX_Enums.h, [1468](#)
- AAX_EModifiers
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_Alt
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_Cntl
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_Command
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_Control
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_None
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_Option
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_SecondaryButton
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_Shift
 - AAX_Enums.h, [1449](#)
- AAX_eModifiers_WINKEY
 - AAX_Enums.h, [1449](#)
- AAX_EndianSwap
 - AAX_EndianSwap.h, [1431](#)
- AAX_EndianSwap.h, [1429](#), [1435](#)
 - AAX_BigEndianNativeSwap, [1432](#)
 - AAX_BigEndianNativeSwapInPlace, [1431](#)
 - AAX_BigEndianNativeSwapSequenceInPlace, [1434](#)
 - AAX_EndianSwap, [1431](#)
 - AAX_EndianSwapInPlace, [1430](#)
 - AAX_EndianSwapSequenceInPlace, [1433](#)
 - AAX_LittleEndianNativeSwap, [1433](#)
 - AAX_LittleEndianNativeSwapInPlace, [1432](#)
 - AAX_LittleEndianNativeSwapSequenceInPlace, [1435](#)
- ENDIANSWAP_H, [1430](#)
- AAX_EndianSwapInPlace
 - AAX_EndianSwap.h, [1430](#)
- AAX_EndianSwapSequenceInPlace
 - AAX_EndianSwap.h, [1433](#)
- AAX_ENotificationEvent
 - AAX_Enums.h, [1457](#)
- AAX_eNotificationEvent_AlgorithmMoved
 - AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_ASAPreviewState
 - AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_ASProcessingState
 - AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_CycleCountChanged
 - AAX_Enums.h, [1459](#)
- AAX_eNotificationEvent_DelayCompensationState
 - AAX_Enums.h, [1459](#)
- AAX_eNotificationEvent_EnteringOfflineMode
 - AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_ExitingOfflineMode
 - AAX_Enums.h, [1459](#)
- AAX_eNotificationEvent_GUIClosed
 - AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_GUIOpened

- AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_HostModeChanged
 - AAX_Enums.h, [1460](#)
- AAX_eNotificationEvent_InsertPositionChanged
 - AAX_Enums.h, [1457](#)
- AAX_eNotificationEvent_LogState
 - AAX_Enums.h, [1461](#)
- AAX_eNotificationEvent_MaxViewSizeChanged
 - AAX_Enums.h, [1459](#)
- AAX_eNotificationEvent_NoiseFloorChanged
 - AAX_Enums.h, [1460](#)
- AAX_eNotificationEvent_ParameterMappingChanged
 - AAX_Enums.h, [1460](#)
- AAX_eNotificationEvent_ParameterNameChanged
 - AAX_Enums.h, [1460](#)
- AAX_eNotificationEvent_PresetOpened
 - AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_PriorSettingsInvalid
 - AAX_Enums.h, [1461](#)
- AAX_eNotificationEvent_SessionBeingOpened
 - AAX_Enums.h, [1458](#)
- AAX_eNotificationEvent_SessionPathChanged
 - AAX_Enums.h, [1459](#)
- AAX_eNotificationEvent_SideChainBeingConnected
 - AAX_Enums.h, [1460](#)
- AAX_eNotificationEvent_SideChainBeingDisconnected
 - AAX_Enums.h, [1460](#)
- AAX_eNotificationEvent_SignalLatencyChanged
 - AAX_Enums.h, [1459](#)
- AAX_eNotificationEvent_TrackNameChanged
 - AAX_Enums.h, [1457](#)
- AAX_eNotificationEvent_TrackPositionChanged
 - AAX_Enums.h, [1457](#)
- AAX_eNotificationEvent_TrackUIDChanged
 - AAX_Enums.h, [1457](#)
- AAX_eNotificationEvent_TransportStateChanged
 - AAX_Enums.h, [1461](#)
- AAX_ENUM_SIZE_CHECK
 - AAX_Enums.h, [1447](#), [1475–1482](#)
 - AAX_Errors.h, [1493](#)
 - AAX_GUITypes.h, [1518](#)
 - AAX_Properties.h, [1623](#)
- AAX_Enums.h, [1437](#), [1482](#)
 - AAE_EAudioBufferLengthNative, [1450](#)
 - AAX_EAssertFlags, [1474](#)
 - AAX_eAssertFlags_Default, [1474](#)
 - AAX_eAssertFlags_Dialog, [1474](#)
 - AAX_eAssertFlags_Log, [1474](#)
 - AAX_EAudioBufferLength, [1449](#)
 - AAX_eAudioBufferLength_1, [1450](#)
 - AAX_eAudioBufferLength_1024, [1450](#)
 - AAX_eAudioBufferLength_128, [1450](#)
 - AAX_eAudioBufferLength_16, [1450](#)
 - AAX_eAudioBufferLength_2, [1450](#)
 - AAX_eAudioBufferLength_256, [1450](#)
 - AAX_eAudioBufferLength_32, [1450](#)
 - AAX_eAudioBufferLength_4, [1450](#)
 - AAX_eAudioBufferLength_512, [1450](#)
 - AAX_eAudioBufferLength_64, [1450](#)
 - AAX_eAudioBufferLength_8, [1450](#)
 - AAX_eAudioBufferLength_Max, [1450](#)
 - AAX_eAudioBufferLength_Undefined, [1450](#)
 - AAX_EAudioBufferLengthDSP, [1450](#)
 - AAX_eAudioBufferLengthDSP_16, [1450](#)
 - AAX_eAudioBufferLengthDSP_32, [1450](#)
 - AAX_eAudioBufferLengthDSP_4, [1450](#)
 - AAX_eAudioBufferLengthDSP_64, [1450](#)
 - AAX_eAudioBufferLengthDSP_Default, [1450](#)
 - AAX_eAudioBufferLengthDSP_Max, [1450](#)
 - AAX_eAudioBufferLengthNative_Max, [1451](#)
 - AAX_eAudioBufferLengthNative_Min, [1451](#)
 - AAX_EComponentInstanceInitAction, [1463](#)
 - AAX_eComponentInstanceInitAction_AddingNewInstance, [1464](#)
 - AAX_eComponentInstanceInitAction_RemovingInstance, [1464](#)
 - AAX_eComponentInstanceInitAction_ResetInstance, [1464](#)
 - AAX_EConstraintLocationMask, [1463](#)
 - AAX_eConstraintLocationMask_DataModel, [1463](#)
 - AAX_eConstraintLocationMask_DLLChipAffinity, [1463](#)
 - AAX_eConstraintLocationMask_None, [1463](#)
 - AAX_EConstraintTopology, [1463](#)
 - AAX_eConstraintTopology_Monolithic, [1463](#)
 - AAX_eConstraintTopology_None, [1463](#)
 - AAX_EDataInPortType, [1469](#)
 - AAX_eDataInPortType_Buffered, [1469](#)
 - AAX_eDataInPortType_Incremental, [1469](#)
 - AAX_eDataInPortType_Unbuffered, [1469](#)
 - AAX_eEQBandType_HighPass, [1466](#)
 - AAX_eEQBandType_HighShelf, [1466](#)
 - AAX_eEQBandType_LowPass, [1466](#)
 - AAX_eEQBandType_LowShelf, [1466](#)
 - AAX_eEQBandType_Notch, [1466](#)
 - AAX_eEQBandType_Parametric, [1466](#)
 - AAX_EEQBandTypes, [1466](#)
 - AAX_EEQInCircuitPolarity, [1467](#)
 - AAX_eEQInCircuitPolarity_Bypassed, [1467](#)
 - AAX_eEQInCircuitPolarity_Disabled, [1467](#)
 - AAX_eEQInCircuitPolarity_Enabled, [1467](#)
 - AAX_EFeetFramesRate, [1470](#)
 - AAX_eFeetFramesRate_23976, [1471](#)
 - AAX_eFeetFramesRate_24, [1471](#)
 - AAX_eFeetFramesRate_25, [1471](#)
 - AAX_EFrameRate, [1470](#)
 - AAX_eFrameRate_100Frame, [1470](#)
 - AAX_eFrameRate_11988DropFrame, [1470](#)
 - AAX_eFrameRate_11988NonDrop, [1470](#)
 - AAX_eFrameRate_120DropFrame, [1470](#)
 - AAX_eFrameRate_120NonDrop, [1470](#)
 - AAX_eFrameRate_23976, [1470](#)
 - AAX_eFrameRate_24Frame, [1470](#)
 - AAX_eFrameRate_25Frame, [1470](#)
 - AAX_eFrameRate_2997DropFrame, [1470](#)
 - AAX_eFrameRate_2997NonDrop, [1470](#)

- AAX_eFrameRate_30DropFrame, [1470](#)
- AAX_eFrameRate_30NonDrop, [1470](#)
- AAX_eFrameRate_47952, [1470](#)
- AAX_eFrameRate_48Frame, [1470](#)
- AAX_eFrameRate_50Frame, [1470](#)
- AAX_eFrameRate_5994DropFrame, [1470](#)
- AAX_eFrameRate_5994NonDrop, [1470](#)
- AAX_eFrameRate_60DropFrame, [1470](#)
- AAX_eFrameRate_60NonDrop, [1470](#)
- AAX_eFrameRate_Undeclared, [1470](#)
- AAX_EHighlightColor, [1448](#)
- AAX_eHighlightColor_Blue, [1448](#)
- AAX_eHighlightColor_Green, [1448](#)
- AAX_eHighlightColor_Num, [1448](#)
- AAX_eHighlightColor_Red, [1448](#)
- AAX_eHighlightColor_Yellow, [1448](#)
- AAX_EHostLevel, [1473](#)
- AAX_eHostLevel_Entry, [1474](#)
- AAX_eHostLevel_Intermediate, [1474](#)
- AAX_eHostLevel_Standard, [1474](#)
- AAX_eHostLevel_Unknown, [1474](#)
- AAX_EHostMode, [1462](#)
- AAX_eHostMode_Config, [1462](#)
- AAX_eHostMode_Show, [1462](#)
- AAX_EHostModeBits, [1461](#)
- AAX_eHostModeBits_Live, [1461](#)
- AAX_eHostModeBits_None, [1461](#)
- AAX_EMaxAudioSuiteTracks, [1451](#)
- AAX_eMaxAudioSuiteTracks, [1451](#)
- AAX_EMeterBallisticType, [1456](#)
- AAX_eMeterBallisticType_Host, [1456](#)
- AAX_eMeterBallisticType_NoDecay, [1456](#)
- AAX_EMeterOrientation, [1455](#)
- AAX_eMeterOrientation_BottomLeft, [1455](#)
- AAX_eMeterOrientation_Center, [1455](#)
- AAX_eMeterOrientation_Default, [1455](#)
- AAX_eMeterOrientation_PhaseDot, [1455](#)
- AAX_eMeterOrientation_TopRight, [1455](#)
- AAX_EMeterType, [1456](#)
- AAX_eMeterType_Analysis, [1456](#)
- AAX_eMeterType_CLGain, [1456](#)
- AAX_eMeterType_EGGain, [1456](#)
- AAX_eMeterType_Input, [1456](#)
- AAX_eMeterType_None, [1456](#)
- AAX_eMeterType_Other, [1456](#)
- AAX_eMeterType_Output, [1456](#)
- AAX_eMIDIBeatClock, [1471](#)
- AAX_eMIDIClick, [1471](#)
- AAX_EMidiGlobalNodeSelectors, [1471](#)
- AAX_eMIDIMtc, [1471](#)
- AAX_EMIDINodeType, [1467](#)
- AAX_eMIDINodeType_Global, [1468](#)
- AAX_eMIDINodeType_LocalInput, [1468](#)
- AAX_eMIDINodeType_LocalOutput, [1468](#)
- AAX_eMIDINodeType_Transport, [1468](#)
- AAX_EModifiers, [1449](#)
- AAX_eModifiers_Alt, [1449](#)
- AAX_eModifiers_Cntl, [1449](#)
- AAX_eModifiers_Command, [1449](#)
- AAX_eModifiers_Control, [1449](#)
- AAX_eModifiers_None, [1449](#)
- AAX_eModifiers_Option, [1449](#)
- AAX_eModifiers_SecondaryButton, [1449](#)
- AAX_eModifiers_Shift, [1449](#)
- AAX_eModifiers_WINKEY, [1449](#)
- AAX_ENotificationEvent, [1457](#)
- AAX_eNotificationEvent_AlgorithmMoved, [1458](#)
- AAX_eNotificationEvent_ASPreviewState, [1458](#)
- AAX_eNotificationEvent_ASProcessingState, [1458](#)
- AAX_eNotificationEvent_CycleCountChanged, [1459](#)
- AAX_eNotificationEvent_DelayCompensationState, [1459](#)
- AAX_eNotificationEvent_EnteringOfflineMode, [1458](#)
- AAX_eNotificationEvent_ExitingOfflineMode, [1459](#)
- AAX_eNotificationEvent_GUIClosed, [1458](#)
- AAX_eNotificationEvent_GUIOpened, [1458](#)
- AAX_eNotificationEvent_HostModeChanged, [1460](#)
- AAX_eNotificationEvent_InsertPositionChanged, [1457](#)
- AAX_eNotificationEvent_LogState, [1461](#)
- AAX_eNotificationEvent_MaxViewSizeChanged, [1459](#)
- AAX_eNotificationEvent_NoiseFloorChanged, [1460](#)
- AAX_eNotificationEvent_ParameterMappingChanged, [1460](#)
- AAX_eNotificationEvent_ParameterNameChanged, [1460](#)
- AAX_eNotificationEvent_PresetOpened, [1458](#)
- AAX_eNotificationEvent_PriorSettingsInvalid, [1461](#)
- AAX_eNotificationEvent_SessionBeingOpened, [1458](#)
- AAX_eNotificationEvent_SessionPathChanged, [1459](#)
- AAX_eNotificationEvent_SideChainBeingConnected, [1460](#)
- AAX_eNotificationEvent_SideChainBeingDisconnected, [1460](#)
- AAX_eNotificationEvent_SignalLatencyChanged, [1459](#)
- AAX_eNotificationEvent_TrackNameChanged, [1457](#)
- AAX_eNotificationEvent_TrackPositionChanged, [1457](#)
- AAX_eNotificationEvent_TrackUIDChanged, [1457](#)
- AAX_eNotificationEvent_TransportStateChanged, [1461](#)
- AAX_ENUM_SIZE_CHECK, [1447](#), [1475–1482](#)
- AAX_ePageTable_EQ_Band_Type, [1466](#)
- AAX_ePageTable_EQ_InCircuitPolarity, [1466](#)
- AAX_ePageTable_UseAlternateControl, [1466](#)
- AAX_EParameterOrientation, [1447](#)
- AAX_eParameterOrientation_BottomMinTopMax, [1465](#)

- AAX_eParameterOrientation_Default, 1465
- AAX_eParameterOrientation_LeftMinRightMax, 1465
- AAX_eParameterOrientation_RightMinLeftMax, 1465
- AAX_eParameterOrientation_RotaryBoostCutMode, 1465
- AAX_eParameterOrientation_RotaryLeftMinRightMax, 1465
- AAX_eParameterOrientation_RotaryRightMinLeftMax, 1465
- AAX_eParameterOrientation_RotarySingleDotMode, 1465
- AAX_eParameterOrientation_RotarySpreadMode, 1465
- AAX_eParameterOrientation_RotaryWrapMode, 1465
- AAX_eParameterOrientation_TopMinBottomMax, 1465
- AAX_EParameterOrientationBits, 1465
- AAX_EParameterType, 1447, 1464
- AAX_eParameterType_Continuous, 1465
- AAX_eParameterType_Discrete, 1465
- AAX_EParameterValueInfoSelector, 1465
- AAX_EPlugInCategory, 1453
- AAX_ePlugInCategory_Delay, 1453
- AAX_ePlugInCategory_Dither, 1453
- AAX_ePlugInCategory_Dynamics, 1453
- AAX_EPlugInCategory_Effect, 1453
- AAX_ePlugInCategory_EQ, 1453
- AAX_ePlugInCategory_Example, 1453
- AAX_ePlugInCategory_Harmonic, 1453
- AAX_ePlugInCategory_HWGenerators, 1453
- AAX_ePlugInCategory_INT32_MAX, 1453
- AAX_EPlugInCategory_MIDIEffect, 1453
- AAX_ePlugInCategory_Modulation, 1453
- AAX_ePlugInCategory_NoiseReduction, 1453
- AAX_ePlugInCategory_None, 1453
- AAX_ePlugInCategory_PitchShift, 1453
- AAX_ePlugInCategory_Reverb, 1453
- AAX_ePlugInCategory_SoundField, 1453
- AAX_ePlugInCategory_SWGenerators, 1453
- AAX_ePlugInCategory_WrappedPlugin, 1453
- AAX_EPlugInStrings, 1454
- AAX_ePlugInStrings_AllSelectedRegionsAnalysis, 1454
- AAX_ePlugInStrings_Analysis, 1454
- AAX_ePlugInStrings_Bypass, 1455
- AAX_ePlugInStrings_ClipName, 1454
- AAX_ePlugInStrings_ClipNameSuffix, 1455
- AAX_ePlugInStrings_INT32_MAX, 1455
- AAX_ePlugInStrings_MonoMode, 1454
- AAX_ePlugInStrings_MultiInputMode, 1454
- AAX_ePlugInStrings_PluginFileName, 1455
- AAX_ePlugInStrings_Preview, 1455
- AAX_ePlugInStrings_Process, 1455
- AAX_ePlugInStrings_Progress, 1455
- AAX_ePlugInStrings_RegionByRegionAnalysis, 1454
- AAX_ePlugInStrings_RegionName, 1454
- AAX_EPreviewState, 1471
- AAX_ePreviewState_Start, 1472
- AAX_ePreviewState_Stop, 1472
- AAX_EPrivateDataOptions, 1462
- AAX_ePrivateDataOptions_Align8, 1462
- AAX_ePrivateDataOptions_DefaultOptions, 1462
- AAX_ePrivateDataOptions_External, 1462
- AAX_ePrivateDataOptions_INT32_MAX, 1462
- AAX_ePrivateDataOptions_KeepOnReset, 1462
- AAX_EProcessingState, 1472
- AAX_eProcessingState_BeginPassGroup, 1472
- AAX_eProcessingState_EndPassGroup, 1472
- AAX_eProcessingState_Start, 1472
- AAX_eProcessingState_StartPass, 1472
- AAX_eProcessingState_Stop, 1472
- AAX_eProcessingState_StopPass, 1472
- AAX_ERecordMode, 1475
- AAX_eRecordMode_Destructive, 1475
- AAX_eRecordMode_None, 1475
- AAX_eRecordMode_Normal, 1475
- AAX_eRecordMode_Num, 1475
- AAX_eRecordMode_QuickPunch, 1475
- AAX_eRecordMode_TrackPunch, 1475
- AAX_eRecordMode_Unknown, 1475
- AAX_EResourceType, 1456
- AAX_eResourceType_None, 1457
- AAX_eResourceType_PageTable, 1457
- AAX_eResourceType_PageTableDir, 1457
- AAX_ESampleRateMask, 1464
- AAX_eSampleRateMask_176400, 1464
- AAX_eSampleRateMask_192000, 1464
- AAX_eSampleRateMask_44100, 1464
- AAX_eSampleRateMask_48000, 1464
- AAX_eSampleRateMask_88200, 1464
- AAX_eSampleRateMask_96000, 1464
- AAX_eSampleRateMask_All, 1464
- AAX_eSampleRateMask_No, 1464
- AAX_EStemFormat, 1451
- AAX_eStemFormat_5_0, 1452
- AAX_eStemFormat_5_0_2, 1452
- AAX_eStemFormat_5_0_4, 1452
- AAX_eStemFormat_5_1, 1452
- AAX_eStemFormat_5_1_2, 1452
- AAX_eStemFormat_5_1_4, 1452
- AAX_eStemFormat_6_0, 1452
- AAX_eStemFormat_6_1, 1452
- AAX_eStemFormat_7_0_2, 1452
- AAX_eStemFormat_7_0_4, 1452
- AAX_eStemFormat_7_0_6, 1452
- AAX_eStemFormat_7_0_DTS, 1452
- AAX_eStemFormat_7_0_SDDS, 1452
- AAX_eStemFormat_7_1_2, 1452
- AAX_eStemFormat_7_1_4, 1452
- AAX_eStemFormat_7_1_6, 1452
- AAX_eStemFormat_7_1_DTS, 1452
- AAX_eStemFormat_7_1_SDDS, 1452

AAX_eStemFormat_9_0_4, [1452](#)
AAX_eStemFormat_9_0_6, [1452](#)
AAX_eStemFormat_9_1_4, [1452](#)
AAX_eStemFormat_9_1_6, [1452](#)
AAX_eStemFormat_Ambi_1_ACN, [1452](#)
AAX_eStemFormat_Ambi_2_ACN, [1452](#)
AAX_eStemFormat_Ambi_3_ACN, [1452](#)
AAX_eStemFormat_Ambi_4_ACN, [1452](#)
AAX_eStemFormat_Ambi_5_ACN, [1452](#)
AAX_eStemFormat_Ambi_6_ACN, [1453](#)
AAX_eStemFormat_Ambi_7_ACN, [1453](#)
AAX_eStemFormat_Any, [1453](#)
AAX_eStemFormat_INT32_MAX, [1453](#)
AAX_eStemFormat_LCR, [1452](#)
AAX_eStemFormat_LCRS, [1452](#)
AAX_eStemFormat_Mono, [1452](#)
AAX_eStemFormat_None, [1453](#)
AAX_eStemFormat_Quad, [1452](#)
AAX_eStemFormat_Stereo, [1452](#)
AAX_eStemFormatNum, [1453](#)
AAX_ESupportLevel, [1473](#)
AAX_eSupportLevel_ByProperty, [1473](#)
AAX_eSupportLevel_Disabled, [1473](#)
AAX_eSupportLevel_Supported, [1473](#)
AAX_eSupportLevel_Uninitialized, [1473](#)
AAX_eSupportLevel_Unsupported, [1473](#)
AAX_ETargetPlatform, [1473](#)
AAX_ETextEncoding, [1474](#)
AAX_eTextEncoding_Num, [1474](#)
AAX_eTextEncoding_Undefined, [1474](#)
AAX_eTextEncoding_UTF8, [1474](#)
AAX_ETracePriorityDSP, [1449](#)
AAX_eTracePriorityDSP_Assert, [1449](#)
AAX_eTracePriorityDSP_High, [1449](#)
AAX_eTracePriorityDSP_Low, [1449](#)
AAX_eTracePriorityDSP_None, [1449](#)
AAX_eTracePriorityDSP_Normal, [1449](#)
AAX_ETracePriorityHost, [1448](#)
AAX_eTracePriorityHost_Critical, [1448](#)
AAX_eTracePriorityHost_High, [1448](#)
AAX_eTracePriorityHost_Low, [1448](#)
AAX_eTracePriorityHost_Lowest, [1448](#)
AAX_eTracePriorityHost_None, [1448](#)
AAX_eTracePriorityHost_Normal, [1448](#)
AAX_ETransportState, [1475](#)
AAX_eTransportState_FastForward, [1475](#)
AAX_eTransportState_Num, [1475](#)
AAX_eTransportState_Paused, [1475](#)
AAX_eTransportState_Play, [1475](#)
AAX_eTransportState_Rewind, [1475](#)
AAX_eTransportState_Scrub, [1475](#)
AAX_eTransportState_Shuttle, [1475](#)
AAX_eTransportState_Stop, [1475](#)
AAX_eTransportState_Stopping, [1475](#)
AAX_eTransportState_Unknown, [1475](#)
AAX_EUpdateSource, [1468](#)
AAX_eUpdateSource_Chunk, [1469](#)
AAX_eUpdateSource_Delay, [1469](#)
AAX_eUpdateSource_Parameter, [1469](#)
AAX_eUpdateSource_Unspecified, [1469](#)
AAX_EUseAlternateControl, [1467](#)
AAX_eUseAlternateControl_No, [1467](#)
AAX_eUseAlternateControl_Yes, [1467](#)
AAX_INT16_MAX, [1446](#)
AAX_INT16_MIN, [1446](#)
AAX_INT32_MAX, [1446](#)
AAX_INT32_MIN, [1446](#)
AAX_STEM_FORMAT, [1447](#)
AAX_STEM_FORMAT_CHANNEL_COUNT, [1447](#)
AAX_STEM_FORMAT_INDEX, [1447](#)
AAX_UINT16_MAX, [1446](#)
AAX_UINT16_MIN, [1446](#)
AAX_UINT32_MAX, [1446](#)
AAX_UINT32_MIN, [1446](#)
kAAX_eTargetPlatform_Count, [1473](#)
kAAX_eTargetPlatform_External, [1473](#)
kAAX_eTargetPlatform_Native, [1473](#)
kAAX_eTargetPlatform_None, [1473](#)
kAAX_eTargetPlatform_TI, [1473](#)
AAX_EnvironmentUtilities.h, [1489](#)
AAX_ePageTable_EQ_Band_Type
AAX_Enums.h, [1466](#)
AAX_ePageTable_EQ_InCircuitPolarity
AAX_Enums.h, [1466](#)
AAX_ePageTable_UseAlternateControl
AAX_Enums.h, [1466](#)
AAX_EParameterOrientation
AAX_Enums.h, [1447](#)
AAX_eParameterOrientation_BottomMinTopMax
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_Default
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_LeftMinRightMax
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_RightMinLeftMax
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_RotaryBoostCutMode
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_RotaryLeftMinRightMax
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_RotaryRightMinLeftMax
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_RotarySingleDotMode
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_RotarySpreadMode
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_RotaryWrapMode
AAX_Enums.h, [1465](#)
AAX_eParameterOrientation_TopMinBottomMax
AAX_Enums.h, [1465](#)
AAX_EParameterOrientationBits
AAX_Enums.h, [1465](#)
AAX_EParameterType
AAX_Enums.h, [1447](#), [1464](#)
AAX_eParameterType_Continuous
AAX_Enums.h, [1465](#)

AAX_eParameterType_Discrete
 AAX_Enums.h, [1465](#)
 AAX_EParameterValueInfoSelector
 AAX_Enums.h, [1465](#)
 AAX_EPluginCategory
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_Delay
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_Dither
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_Dynamics
 AAX_Enums.h, [1453](#)
 AAX_EPluginCategory_Effect
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_EQ
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_Example
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_Harmonic
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_HWGenerators
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_INT32_MAX
 AAX_Enums.h, [1453](#)
 AAX_EPluginCategory_MIDIEffect
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_Modulation
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_NoiseReduction
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_None
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_PitchShift
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_Reverb
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_SoundField
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_SWGenerators
 AAX_Enums.h, [1453](#)
 AAX_ePluginCategory_WrappedPlugin
 AAX_Enums.h, [1453](#)
 AAX_EPluginStrings
 AAX_Enums.h, [1454](#)
 AAX_ePluginStrings_AllSelectedRegionsAnalysis
 AAX_Enums.h, [1454](#)
 AAX_ePluginStrings_Analysis
 AAX_Enums.h, [1454](#)
 AAX_ePluginStrings_Bypass
 AAX_Enums.h, [1455](#)
 AAX_ePluginStrings_ClipName
 AAX_Enums.h, [1454](#)
 AAX_ePluginStrings_ClipNameSuffix
 AAX_Enums.h, [1455](#)
 AAX_ePluginStrings_INT32_MAX
 AAX_Enums.h, [1455](#)
 AAX_ePluginStrings_MonoMode
 AAX_Enums.h, [1454](#)
 AAX_ePluginStrings_MultiInputMode
 AAX_Enums.h, [1454](#)
 AAX_ePluginStrings_PluginFileName
 AAX_Enums.h, [1455](#)
 AAX_ePluginStrings_Preview
 AAX_Enums.h, [1455](#)
 AAX_ePluginStrings_Process
 AAX_Enums.h, [1455](#)
 AAX_ePluginStrings_Progress
 AAX_Enums.h, [1455](#)
 AAX_ePluginStrings_RegionByRegionAnalysis
 AAX_Enums.h, [1454](#)
 AAX_ePluginStrings_RegionName
 AAX_Enums.h, [1454](#)
 AAX_EPreviewState
 AAX_Enums.h, [1471](#)
 AAX_ePreviewState_Start
 AAX_Enums.h, [1472](#)
 AAX_ePreviewState_Stop
 AAX_Enums.h, [1472](#)
 AAX_EPrivateDataOptions
 AAX_Enums.h, [1462](#)
 AAX_ePrivateDataOptions_Align8
 AAX_Enums.h, [1462](#)
 AAX_ePrivateDataOptions_DefaultOptions
 AAX_Enums.h, [1462](#)
 AAX_ePrivateDataOptions_External
 AAX_Enums.h, [1462](#)
 AAX_ePrivateDataOptions_INT32_MAX
 AAX_Enums.h, [1462](#)
 AAX_ePrivateDataOptions_KeepOnReset
 AAX_Enums.h, [1462](#)
 AAX_EProcessingState
 AAX_Enums.h, [1472](#)
 AAX_eProcessingState_BeginPassGroup
 AAX_Enums.h, [1472](#)
 AAX_eProcessingState_EndPassGroup
 AAX_Enums.h, [1472](#)
 AAX_eProcessingState_Start
 AAX_Enums.h, [1472](#)
 AAX_eProcessingState_StartPass
 AAX_Enums.h, [1472](#)
 AAX_eProcessingState_Stop
 AAX_Enums.h, [1472](#)
 AAX_eProcessingState_StopPass
 AAX_Enums.h, [1472](#)
 AAX_EProperty
 AAX_Properties.h, [1605](#)
 AAX_eProperty_AllowPreviewWithoutAnalysis
 AAX_Properties.h, [1615](#)
 AAX_eProperty_AlwaysBypass
 AAX_Properties.h, [1614](#)
 AAX_eProperty_AudioBufferLength
 AAX_Properties.h, [1611](#)
 AAX_eProperty_AudiosuitePropsBase
 AAX_Properties.h, [1614](#)
 AAX_eProperty_CanBypass
 AAX_Properties.h, [1613](#)

- AAX_eProperty_Constraint_AlwaysProcess
AAX_Properties.h, [1622](#)
- AAX_eProperty_Constraint_DoNotApplyDefaultSettings
AAX_Properties.h, [1622](#)
- AAX_eProperty_Constraint_Location
AAX_Properties.h, [1619](#)
- AAX_eProperty_Constraint_MultiMonoSupport
AAX_Properties.h, [1620](#)
- AAX_eProperty_Constraint_NeverCache
AAX_Properties.h, [1620](#)
- AAX_eProperty_Constraint_NeverUnload
AAX_Properties.h, [1619](#)
- AAX_eProperty_Constraint_Topology
AAX_Properties.h, [1619](#)
- AAX_eProperty_ConstraintBase
AAX_Properties.h, [1619](#)
- AAX_eProperty_ConstraintBase_2
AAX_Properties.h, [1621](#)
- AAX_eProperty_ContinuousOnly
AAX_Properties.h, [1616](#)
- AAX_eProperty_DebugPropertiesBase
AAX_Properties.h, [1622](#)
- AAX_eProperty_Deprecated_DSP_Plugin_List
AAX_Properties.h, [1609](#)
- AAX_eProperty_Deprecated_Native_Plugin_List
AAX_Properties.h, [1610](#)
- AAX_eProperty_Deprecated_Plugin_List
AAX_Properties.h, [1609](#)
- AAX_eProperty_DestinationTrack
AAX_Properties.h, [1616](#)
- AAX_eProperty_DisableAudiosuiteReverse
AAX_Properties.h, [1618](#)
- AAX_eProperty_DisableHandles
AAX_Properties.h, [1617](#)
- AAX_eProperty_DisablePreview
AAX_Properties.h, [1617](#)
- AAX_eProperty_DoesntIncrOutputSample
AAX_Properties.h, [1617](#)
- AAX_eProperty_DSP_AudioBufferLength
AAX_Properties.h, [1611](#)
- AAX_eProperty_EnableHostDebugLogs
AAX_Properties.h, [1623](#)
- AAX_eProperty_ExternalProcessorTypeID
AAX_Properties.h, [1610](#)
- AAX_eProperty_FeaturesBase
AAX_Properties.h, [1620](#)
- AAX_eProperty_GeneralPropsBase
AAX_Properties.h, [1611](#)
- AAX_eProperty_GUIBase
AAX_Properties.h, [1618](#)
- AAX_eProperty_HybridInputStemFormat
AAX_Properties.h, [1614](#)
- AAX_eProperty_HybridOutputStemFormat
AAX_Properties.h, [1614](#)
- AAX_eProperty_InputStemFormat
AAX_Properties.h, [1611](#)
- AAX_eProperty_LatencyContribution
AAX_Properties.h, [1612](#)
- AAX_eProperty_ManufacturerID
AAX_Properties.h, [1606](#)
- AAX_eProperty_MaxASProp
AAX_Properties.h, [1618](#)
- AAX_eProperty_MaxCap
AAX_Properties.h, [1623](#)
- AAX_eProperty_MaxConstraintProp
AAX_Properties.h, [1620](#)
- AAX_eProperty_MaxConstraintProp_2
AAX_Properties.h, [1622](#)
- AAX_eProperty_MaxFeaturesProp
AAX_Properties.h, [1621](#)
- AAX_eProperty_MaxGUIProp
AAX_Properties.h, [1618](#)
- AAX_eProperty_MaxMeterProp
AAX_Properties.h, [1619](#)
- AAX_eProperty_MaxProp
AAX_Properties.h, [1623](#)
- AAX_eProperty_Meter_Ballistics
AAX_Properties.h, [1619](#)
- AAX_eProperty_Meter_Orientation
AAX_Properties.h, [1619](#)
- AAX_eProperty_Meter_Type
AAX_Properties.h, [1619](#)
- AAX_eProperty_MeterBase
AAX_Properties.h, [1618](#)
- AAX_eProperty_MinProp
AAX_Properties.h, [1606](#)
- AAX_eProperty_MultiInputModeOnly
AAX_Properties.h, [1616](#)
- AAX_eProperty_NativeBackgroundProc
AAX_Properties.h, [1611](#)
- AAX_eProperty_NativeInstanceInitProc
AAX_Properties.h, [1610](#)
- AAX_eProperty_NativeProcessProc
AAX_Properties.h, [1610](#)
- AAX_eProperty_NeedsOutputDithered
AAX_Properties.h, [1618](#)
- AAX_eProperty_NoID
AAX_Properties.h, [1606](#)
- AAX_eProperty_NumberOfInputs
AAX_Properties.h, [1617](#)
- AAX_eProperty_NumberOfOutputs
AAX_Properties.h, [1617](#)
- AAX_eProperty_ObservesTransportState
AAX_Properties.h, [1621](#)
- AAX_eProperty_OptionalAnalysis
AAX_Properties.h, [1615](#)
- AAX_eProperty_OutputStemFormat
AAX_Properties.h, [1611](#)
- AAX_eProperty_PluginID_AudioSuite
AAX_Properties.h, [1607](#)
- AAX_eProperty_PluginID_Deprecated
AAX_Properties.h, [1609](#)
- AAX_eProperty_PluginID_ExternalProcessor
AAX_Properties.h, [1610](#)
- AAX_eProperty_PluginID_Native
AAX_Properties.h, [1607](#)

- AAX_eProperty_PluginID_NoProcessing
AAX_Properties.h, [1608](#)
- AAX_eProperty_PluginID_RTAS
AAX_Properties.h, [1607](#)
- AAX_eProperty_PluginID_TI
AAX_Properties.h, [1608](#)
- AAX_eProperty_PluginSpecPropsBase
AAX_Properties.h, [1606](#)
- AAX_eProperty_ProcessProcPropsBase
AAX_Properties.h, [1610](#)
- AAX_eProperty_ProductID
AAX_Properties.h, [1606](#)
- AAX_eProperty_Related_DSP_Plugin_List
AAX_Properties.h, [1609](#)
- AAX_eProperty_Related_Native_Plugin_List
AAX_Properties.h, [1609](#)
- AAX_eProperty_RequestsAllTrackData
AAX_Properties.h, [1616](#)
- AAX_eProperty_RequiresAnalysis
AAX_Properties.h, [1615](#)
- AAX_eProperty_RequiresChunkCallsOnMainThread
AAX_Properties.h, [1621](#)
- AAX_eProperty_SampleRate
AAX_Properties.h, [1612](#)
- AAX_eProperty_ShowInMenus
AAX_Properties.h, [1614](#)
- AAX_eProperty_SideChainStemFormat
AAX_Properties.h, [1613](#)
- AAX_eProperty_StoreXMLPageTablesByEffect
AAX_Properties.h, [1621](#)
- AAX_eProperty_StoreXMLPageTablesByType
AAX_Properties.h, [1621](#)
- AAX_eProperty_SupportsSaveRestore
AAX_Properties.h, [1620](#)
- AAX_eProperty_SupportsSideChainInput
AAX_Properties.h, [1618](#)
- AAX_eProperty_TI_ForceAllowChipSharing
AAX_Properties.h, [1614](#)
- AAX_eProperty_TI_InstanceCycleCount
AAX_Properties.h, [1613](#)
- AAX_eProperty_TI_MaxInstancesPerChip
AAX_Properties.h, [1613](#)
- AAX_eProperty_TI_SharedCycleCount
AAX_Properties.h, [1613](#)
- AAX_eProperty_TIBackgroundProc
AAX_Properties.h, [1611](#)
- AAX_eProperty_TIDLLFileName
AAX_Properties.h, [1611](#)
- AAX_eProperty_TIInstanceInitProc
AAX_Properties.h, [1611](#)
- AAX_eProperty_TIProcessProc
AAX_Properties.h, [1611](#)
- AAX_eProperty_UsesClientGUI
AAX_Properties.h, [1618](#)
- AAX_eProperty_UsesRandomAccess
AAX_Properties.h, [1615](#)
- AAX_eProperty_UsesTransport
AAX_Properties.h, [1620](#)
- AAX_eProperty_UsesTransportControl
AAX_Properties.h, [1621](#)
- AAX_ERecordMode
AAX_Enums.h, [1475](#)
- AAX_eRecordMode_Destructive
AAX_Enums.h, [1475](#)
- AAX_eRecordMode_None
AAX_Enums.h, [1475](#)
- AAX_eRecordMode_Normal
AAX_Enums.h, [1475](#)
- AAX_eRecordMode_Num
AAX_Enums.h, [1475](#)
- AAX_eRecordMode_QuickPunch
AAX_Enums.h, [1475](#)
- AAX_eRecordMode_TrackPunch
AAX_Enums.h, [1475](#)
- AAX_eRecordMode_Unknown
AAX_Enums.h, [1475](#)
- AAX_EResourceType
AAX_Enums.h, [1456](#)
- AAX_eResourceType_None
AAX_Enums.h, [1457](#)
- AAX_eResourceType_PageTable
AAX_Enums.h, [1457](#)
- AAX_eResourceType_PageTableDir
AAX_Enums.h, [1457](#)
- AAX_ERROR_ACF_ERROR
AAX_Errors.h, [1492](#)
- AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW
AAX_Errors.h, [1493](#)
- AAX_ERROR_ARGUMENT_OUT_OF_RANGE
AAX_Errors.h, [1493](#)
- AAX_ERROR_CONTEXT_ALREADY_HAS_METERS
AAX_Errors.h, [1492](#)
- AAX_ERROR_DIRECT_ACCESS_OUT_OF_BOUNDS
AAX_Errors.h, [1492](#)
- AAX_ERROR_DUPLICATE_EFFECT_ID
AAX_Errors.h, [1492](#)
- AAX_ERROR_DUPLICATE_TYPE_ID
AAX_Errors.h, [1492](#)
- AAX_ERROR_EMPTY_EFFECT_NAME
AAX_Errors.h, [1492](#)
- AAX_ERROR_FIELD_TYPE_DOES_NOT_SUPPORT_DIRECT_ACCESS
AAX_Errors.h, [1492](#)
- AAX_ERROR_FIFO_FULL
AAX_Errors.h, [1492](#)
- AAX_ERROR_INCORRECT_CHUNK_SIZE
AAX_Errors.h, [1492](#)
- AAX_ERROR_INITIALIZING_PACKET_STREAM_THREAD
AAX_Errors.h, [1492](#)
- AAX_ERROR_INVALID_ARGUMENT
AAX_Errors.h, [1493](#)
- AAX_ERROR_INVALID_CHUNK_ID
AAX_Errors.h, [1492](#)
- AAX_ERROR_INVALID_CHUNK_INDEX
AAX_Errors.h, [1492](#)
- AAX_ERROR_INVALID_FIELD_INDEX
AAX_Errors.h, [1492](#)

AAX_ERROR_INVALID_INTERNAL_DATA
AAX_Errors.h, [1493](#)

AAX_ERROR_INVALID_METER_INDEX
AAX_Errors.h, [1492](#)

AAX_ERROR_INVALID_METER_TYPE
AAX_Errors.h, [1492](#)

AAX_ERROR_INVALID_PARAMETER_ID
AAX_Errors.h, [1492](#)

AAX_ERROR_INVALID_PARAMETER_INDEX
AAX_Errors.h, [1492](#)

AAX_ERROR_INVALID_PATH
AAX_Errors.h, [1492](#)

AAX_ERROR_INVALID_STRING_CONVERSION
AAX_Errors.h, [1492](#)

AAX_ERROR_INVALID_VIEW_SIZE
AAX_Errors.h, [1492](#)

AAX_ERROR_MALFORMED_CHUNK
AAX_Errors.h, [1492](#)

AAX_ERROR_MIXER_THREAD_FALLING_BEHIND
AAX_Errors.h, [1492](#)

AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME
AAX_Errors.h, [1493](#)

AAX_ERROR_NO_COMPONENTS
AAX_Errors.h, [1492](#)

AAX_ERROR_NOT_INITIALIZED
AAX_Errors.h, [1492](#)

AAX_ERROR_NOTIFICATION_FAILED
AAX_Errors.h, [1492](#)

AAX_ERROR_NULL_ARGUMENT
AAX_Errors.h, [1493](#)

AAX_ERROR_NULL_COMPONENT
AAX_Errors.h, [1492](#)

AAX_ERROR_NULL_OBJECT
AAX_Errors.h, [1492](#)

AAX_ERROR_OLDER_VERSION
AAX_Errors.h, [1492](#)

AAX_ERROR_PLUGIN_BEGIN
AAX_Errors.h, [1493](#)

AAX_ERROR_PLUGIN_END
AAX_Errors.h, [1493](#)

AAX_ERROR_PLUGIN_NOT_AUTHORIZED
AAX_Errors.h, [1492](#)

AAX_ERROR_PLUGIN_NULL_PARAMETER
AAX_Errors.h, [1492](#)

AAX_ERROR_PORT_ID_OUT_OF_RANGE
AAX_Errors.h, [1492](#)

AAX_ERROR_POST_PACKET_FAILED
AAX_Errors.h, [1492](#)

AAX_ERROR_PRINT_FAILURE
AAX_Errors.h, [1493](#)

AAX_ERROR_PROPERTY_UNDEFINED
AAX_Errors.h, [1492](#)

AAX_ERROR_SIGNED_INT_OVERFLOW
AAX_Errors.h, [1492](#)

AAX_ERROR_TOD_BEHIND
AAX_Errors.h, [1492](#)

AAX_ERROR_UNEXPECTED_EFFECT_ID
AAX_Errors.h, [1493](#)

AAX_ERROR_UNIMPLEMENTED
AAX_Errors.h, [1492](#)

AAX_ERROR_UNKNOWN_EXCEPTION
AAX_Errors.h, [1492](#)

AAX_ERROR_UNKNOWN_ID
AAX_Errors.h, [1492](#)

AAX_ERROR_UNKNOWN_PLUGIN
AAX_Errors.h, [1492](#)

AAX_ERROR_UNSUPPORTED_ENCODING
AAX_Errors.h, [1493](#)

AAX_Errors.h, [1490](#), [1493](#)
AAX_EError, [1491](#)
AAX_ENUM_SIZE_CHECK, [1493](#)
AAX_ERROR_ACF_ERROR, [1492](#)
AAX_ERROR_ARGUMENT_BUFFER_OVERFLOW,
[1493](#)
AAX_ERROR_ARGUMENT_OUT_OF_RANGE,
[1493](#)
AAX_ERROR_CONTEXT_ALREADY_HAS_METERS,
[1492](#)
AAX_ERROR_DIRECT_ACCESS_OUT_OF_BOUNDS,
[1492](#)
AAX_ERROR_DUPLICATE_EFFECT_ID, [1492](#)
AAX_ERROR_DUPLICATE_TYPE_ID, [1492](#)
AAX_ERROR_EMPTY_EFFECT_NAME, [1492](#)
AAX_ERROR_FIELD_TYPE_DOES_NOT_SUPPORT_DIRECT_ACCESS,
[1492](#)
AAX_ERROR_FIFO_FULL, [1492](#)
AAX_ERROR_INCORRECT_CHUNK_SIZE, [1492](#)
AAX_ERROR_INITIALIZING_PACKET_STREAM_THREAD,
[1492](#)
AAX_ERROR_INVALID_ARGUMENT, [1493](#)
AAX_ERROR_INVALID_CHUNK_ID, [1492](#)
AAX_ERROR_INVALID_CHUNK_INDEX, [1492](#)
AAX_ERROR_INVALID_FIELD_INDEX, [1492](#)
AAX_ERROR_INVALID_INTERNAL_DATA, [1493](#)
AAX_ERROR_INVALID_METER_INDEX, [1492](#)
AAX_ERROR_INVALID_METER_TYPE, [1492](#)
AAX_ERROR_INVALID_PARAMETER_ID, [1492](#)
AAX_ERROR_INVALID_PARAMETER_INDEX,
[1492](#)
AAX_ERROR_INVALID_PATH, [1492](#)
AAX_ERROR_INVALID_STRING_CONVERSION,
[1492](#)
AAX_ERROR_INVALID_VIEW_SIZE, [1492](#)
AAX_ERROR_MALFORMED_CHUNK, [1492](#)
AAX_ERROR_MIXER_THREAD_FALLING_BEHIND,
[1492](#)
AAX_ERROR_NO_ABBREVIATED_PARAMETER_NAME,
[1493](#)
AAX_ERROR_NO_COMPONENTS, [1492](#)
AAX_ERROR_NOT_INITIALIZED, [1492](#)
AAX_ERROR_NOTIFICATION_FAILED, [1492](#)
AAX_ERROR_NULL_ARGUMENT, [1493](#)
AAX_ERROR_NULL_COMPONENT, [1492](#)
AAX_ERROR_NULL_OBJECT, [1492](#)
AAX_ERROR_OLDER_VERSION, [1492](#)
AAX_ERROR_PLUGIN_BEGIN, [1493](#)

AAX_ERROR_PLUGIN_END, [1493](#)
 AAX_ERROR_PLUGIN_NOT_AUTHORIZED, [1492](#)
 AAX_ERROR_PLUGIN_NULL_PARAMETER, [1492](#)
 AAX_ERROR_PORT_ID_OUT_OF_RANGE, [1492](#)
 AAX_ERROR_POST_PACKET_FAILED, [1492](#)
 AAX_ERROR_PRINT_FAILURE, [1493](#)
 AAX_ERROR_PROPERTY_UNDEFINED, [1492](#)
 AAX_ERROR_SIGNED_INT_OVERFLOW, [1492](#)
 AAX_ERROR_TOD_BEHIND, [1492](#)
 AAX_ERROR_UNEXPECTED_EFFECT_ID, [1493](#)
 AAX_ERROR_UNIMPLEMENTED, [1492](#)
 AAX_ERROR_UNKNOWN_EXCEPTION, [1492](#)
 AAX_ERROR_UNKNOWN_ID, [1492](#)
 AAX_ERROR_UNKNOWN_PLUGIN, [1492](#)
 AAX_ERROR_UNSUPPORTED_ENCODING, [1493](#)
 AAX_RESULT_ADD_FIELD_UNSUPPORTED_FIELD, [1492](#)
 AAX_RESULT_NEW_PACKET_POSTED, [1492](#)
 AAX_RESULT_PACKET_STREAM_NOT_EMPTY, [1492](#)
 AAX_SUCCESS, [1492](#)
 AAX_ESampleRateMask
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_176400
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_192000
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_44100
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_48000
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_88200
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_96000
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_All
 AAX_Enums.h, [1464](#)
 AAX_eSampleRateMask_No
 AAX_Enums.h, [1464](#)
 AAX_EStemFormat
 AAX_Enums.h, [1451](#)
 AAX_eStemFormat_5_0
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_5_0_2
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_5_0_4
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_5_1
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_5_1_2
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_5_1_4
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_6_0
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_6_1
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_0_2
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_0_4
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_0_6
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_0_DTS
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_0_SDDS
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_1_2
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_1_4
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_1_6
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_1_DTS
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_7_1_SDDS
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_9_0_4
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_9_0_6
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_9_1_4
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_9_1_6
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_Ambi_1_ACN
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_Ambi_2_ACN
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_Ambi_3_ACN
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_Ambi_4_ACN
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_Ambi_5_ACN
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_Ambi_6_ACN
 AAX_Enums.h, [1453](#)
 AAX_eStemFormat_Ambi_7_ACN
 AAX_Enums.h, [1453](#)
 AAX_eStemFormat_Any
 AAX_Enums.h, [1453](#)
 AAX_eStemFormat_INT32_MAX
 AAX_Enums.h, [1453](#)
 AAX_eStemFormat_LCR
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_LCRS
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_Mono
 AAX_Enums.h, [1452](#)
 AAX_eStemFormat_None
 AAX_Enums.h, [1453](#)
 AAX_eStemFormat_Quad
 AAX_Enums.h, [1452](#)

AAX_eStemFormat_Stereo
AAX_Enums.h, [1452](#)

AAX_eStemFormatNum
AAX_Enums.h, [1453](#)

AAX_ESupportLevel
AAX_Enums.h, [1473](#)

AAX_eSupportLevel_ByProperty
AAX_Enums.h, [1473](#)

AAX_eSupportLevel_Disabled
AAX_Enums.h, [1473](#)

AAX_eSupportLevel_Supported
AAX_Enums.h, [1473](#)

AAX_eSupportLevel_Uninitialized
AAX_Enums.h, [1473](#)

AAX_eSupportLevel_Unsupported
AAX_Enums.h, [1473](#)

AAX_ETargetPlatform
AAX_Enums.h, [1473](#)

AAX_ETextEncoding
AAX_Enums.h, [1474](#)

AAX_eTextEncoding_Num
AAX_Enums.h, [1474](#)

AAX_eTextEncoding_Undefined
AAX_Enums.h, [1474](#)

AAX_eTextEncoding_UTF8
AAX_Enums.h, [1474](#)

AAX_ETracePriority
AAX_Assert.h, [1329](#)

AAX_ETracePriorityDSP
AAX_Enums.h, [1449](#)

AAX_eTracePriorityDSP_Assert
AAX_Enums.h, [1449](#)

AAX_eTracePriorityDSP_High
AAX_Enums.h, [1449](#)

AAX_eTracePriorityDSP_Low
AAX_Enums.h, [1449](#)

AAX_eTracePriorityDSP_None
AAX_Enums.h, [1449](#)

AAX_eTracePriorityDSP_Normal
AAX_Enums.h, [1449](#)

AAX_ETracePriorityHost
AAX_Enums.h, [1448](#)

AAX_eTracePriorityHost_Critical
AAX_Enums.h, [1448](#)

AAX_eTracePriorityHost_High
AAX_Enums.h, [1448](#)

AAX_eTracePriorityHost_Low
AAX_Enums.h, [1448](#)

AAX_eTracePriorityHost_Lowest
AAX_Enums.h, [1448](#)

AAX_eTracePriorityHost_None
AAX_Enums.h, [1448](#)

AAX_eTracePriorityHost_Normal
AAX_Enums.h, [1448](#)

AAX_ETransportState
AAX_Enums.h, [1475](#)

AAX_eTransportState_FastForward
AAX_Enums.h, [1475](#)

AAX_eTransportState_Num
AAX_Enums.h, [1475](#)

AAX_eTransportState_Paused
AAX_Enums.h, [1475](#)

AAX_eTransportState_Play
AAX_Enums.h, [1475](#)

AAX_eTransportState_Rewind
AAX_Enums.h, [1475](#)

AAX_eTransportState_Scrub
AAX_Enums.h, [1475](#)

AAX_eTransportState_Shuttle
AAX_Enums.h, [1475](#)

AAX_eTransportState_Stop
AAX_Enums.h, [1475](#)

AAX_eTransportState_Stopping
AAX_Enums.h, [1475](#)

AAX_eTransportState_Unknown
AAX_Enums.h, [1475](#)

AAX_EUpdateSource
AAX_Enums.h, [1468](#)

AAX_eUpdateSource_Chunk
AAX_Enums.h, [1469](#)

AAX_eUpdateSource_Delay
AAX_Enums.h, [1469](#)

AAX_eUpdateSource_Parameter
AAX_Enums.h, [1469](#)

AAX_eUpdateSource_Unspecified
AAX_Enums.h, [1469](#)

AAX_EUseAlternateControl
AAX_Enums.h, [1467](#)

AAX_eUseAlternateControl_No
AAX_Enums.h, [1467](#)

AAX_eUseAlternateControl_Yes
AAX_Enums.h, [1467](#)

AAX_EViewContainer_Type
AAX_GUITypes.h, [1515](#), [1516](#)

AAX_eViewContainer_Type_HWND
AAX_GUITypes.h, [1516](#)

AAX_eViewContainer_Type_NSView
AAX_GUITypes.h, [1516](#)

AAX_eViewContainer_Type_NULL
AAX_GUITypes.h, [1516](#)

AAX_eViewContainer_Type_UIView
AAX_GUITypes.h, [1516](#)

AAX_Exception.h, [1503](#), [1506](#)

AAX_CAPTURE, [1504](#)

AAX_CAPTURE_MULT, [1505](#)

AAX_SWALLOW, [1504](#)

AAX_SWALLOW_MULT, [1504](#)

AAX_EXPORT
AAX_Exports.cpp, [1510](#)

AAX_Exports.cpp, [1510](#)

AAX_EXPORT, [1510](#)

ACFCanUnloadNow, [1512](#)

ACFGetClassFactory, [1511](#)

ACFGetSDKVersion, [1513](#)

ACFRegisterComponent, [1511](#)

ACFRegisterPlugin, [1511](#)

- ACFSShutdown, [1513](#)
- ACFStartup, [1512](#)
- AAX_FastInterpolatedTableLookup< TFloat, DFloat
>, [740](#)
 - DoTableLookupExtraFast, [741](#), [742](#)
 - DoTableLookupExtraFastMulti, [742](#)
 - GetMaxMinusMin, [742](#)
 - GetMin, [742](#)
 - SetParameters, [741](#)
- AAX_FastInterpolatedTableLookup.h, [1707](#), [1708](#)
 - AAX_FASTINTERPOLATEDTABLELOOKUP_H,
[1707](#)
- AAX_FastInterpolatedTableLookup.hpp, [1709](#)
- AAX_FASTINTERPOLATEDTABLELOOKUP_H
 - AAX_FastInterpolatedTableLookup.h, [1707](#)
- AAX_FastPow.h, [1710](#), [1711](#)
 - _AAX_FASTPOW_H_, [1711](#)
- AAX_Feature_UID
 - AAX.h, [1315](#)
 - AAX_UIDs.h, [1654](#)
- AAX_FIELD_INDEX
 - AAX.h, [1311](#)
- AAX_FINAL
 - AAX.h, [1307](#)
- AAX_Getting_Started_Guide.doxygen, [1295](#)
- AAX_GUITypes.h, [1514](#), [1518](#)
 - AAX_ENUM_SIZE_CHECK, [1518](#)
 - AAX_EViewContainer_Type, [1515](#), [1516](#)
 - AAX_eViewContainer_Type_HWND, [1516](#)
 - AAX_eViewContainer_Type_NSView, [1516](#)
 - AAX_eViewContainer_Type_NULL, [1516](#)
 - AAX_eViewContainer_Type_UIView, [1516](#)
 - AAX_Point, [1515](#)
 - AAX_Rect, [1515](#)
 - operator!=, [1516](#), [1517](#)
 - operator<, [1516](#)
 - operator<=, [1517](#)
 - operator>, [1517](#)
 - operator>=, [1517](#)
 - operator==, [1516](#), [1517](#)
- AAX_HI
 - AAX_MiscUtils.h, [1716](#)
- AAX_HostSupport.doxygen, [1295](#)
- AAX_IACFAutomationDelegate, [743](#)
 - GetTouchState, [746](#)
 - PostCurrentValue, [745](#)
 - PostReleaseRequest, [746](#)
 - PostSetValueRequest, [745](#)
 - PostTouchRequest, [745](#)
 - RegisterParameter, [744](#)
 - UnregisterParameter, [744](#)
- AAX_IACFAutomationDelegate.h, [1519](#), [1520](#)
- AAX_IACFCollection, [746](#)
 - AddEffect, [748](#)
 - AddPackageName, [748](#)
 - SetManufacturerName, [748](#)
 - SetPackageVersion, [749](#)
 - SetProperties, [749](#)
- AAX_IACFCollection.h, [1521](#)
- AAX_IACFComponentDescriptor, [749](#)
 - AddAudioBufferLength, [753](#)
 - AddAudioIn, [752](#)
 - AddAudioOut, [752](#)
 - AddAuxOutputStem, [755](#)
 - AddClock, [754](#)
 - AddDataInPort, [754](#)
 - AddDmaInstance, [756](#)
 - AddMeters, [759](#)
 - AddMIDINode, [757](#)
 - AddPrivateData, [756](#)
 - AddProcessProc_Native, [757](#)
 - AddProcessProc_TI, [758](#)
 - AddReservedField, [752](#)
 - AddSampleRate, [753](#)
 - AddSideChainIn, [754](#)
 - Clear, [751](#)
- AAX_IACFComponentDescriptor.h, [1522](#)
- AAX_IACFComponentDescriptor_V2, [759](#)
 - AddTemporaryData, [762](#)
- AAX_IACFComponentDescriptor_V3, [762](#)
 - AddProcessProc, [765](#)
- AAX_IACFController, [766](#)
 - ClearMeterClipped, [774](#)
 - ClearMeterPeakValue, [773](#)
 - GetCurrentMeterValue, [772](#)
 - GetCycleCount, [769](#)
 - GetEffectID, [768](#)
 - GetInputStemFormat, [768](#)
 - GetMeterClipped, [773](#)
 - GetMeterCount, [774](#)
 - GetMeterPeakValue, [773](#)
 - GetNextMIDIpacket, [774](#)
 - GetOutputStemFormat, [769](#)
 - GetSampleRate, [768](#)
 - GetSignalLatency, [769](#)
 - GetTODLocation, [770](#)
 - PostPacket, [772](#)
 - SetCycleCount, [771](#)
 - SetSignalLatency, [771](#)
- AAX_IACFController.h, [1523](#), [1524](#)
- AAX_IACFController_V2, [775](#)
 - GetCurrentAutomationTimestamp, [778](#)
 - GetHostName, [779](#)
 - GetHybridSignalLatency, [778](#)
 - SendNotification, [777](#)
- AAX_IACFController_V3, [779](#)
 - GetIsAudioSuite, [782](#)
 - GetPlugInTargetPlatform, [782](#)
- AAX_IACFDataBuffer, [783](#)
 - Data, [784](#)
 - Size, [784](#)
 - Type, [784](#)
- AAX_IACFDataBuffer.h, [1526](#), [1527](#)
 - AAX_IACFDataBuffer_H, [1526](#)
- AAX_IACFDataBuffer_H
 - AAX_IACFDataBuffer.h, [1526](#)

- AAX_IACFDescriptionHost, 785
 - AcquireFeatureProperties, 786
- AAX_IACFDescriptionHost.h, 1527
- AAX_IACFEffectDescriptor, 787
 - AddCategory, 789
 - AddCategoryBypassParameter, 789
 - AddComponent, 788
 - AddMeterDescription, 791
 - AddName, 788
 - AddProcPtr, 789
 - AddResourceInfo, 791
 - SetProperties, 791
- AAX_IACFEffectDescriptor.h, 1528
- AAX_IACFEffectDescriptor_V2, 792
 - AddControlMIDINode, 793
- AAX_IACFEffectDirectData, 794
 - Initialize, 796
 - TimerWakeup, 796
 - Uninitialize, 796
- AAX_IACFEffectDirectData.h, 1529, 1530
- AAX_IACFEffectDirectData_V2, 797
 - NotificationReceived, 798
- AAX_IACFEffectGUI, 799
 - Draw, 804
 - GetCustomLabel, 805
 - GetViewSize, 803
 - Initialize, 802
 - NotificationReceived, 802
 - ParameterUpdated, 804
 - SetControlHighlightInfo, 805
 - SetViewContainer, 803
 - TimerWakeup, 804
 - Uninitialize, 802
- AAX_IACFEffectGUI.h, 1530, 1531
- AAX_IACFEffectParameters, 806
 - CompareActiveChunk, 828
 - DoMIDITransfers, 830
 - GenerateCoefficients, 825
 - GetChunk, 827
 - GetChunkIDFromIndex, 826
 - GetChunkSize, 826
 - GetCustomData, 829
 - GetMasterBypassParameter, 813
 - GetNumberOfChanges, 828
 - GetNumberOfChunks, 826
 - GetNumberOfParameters, 813
 - GetParameter, 817
 - GetParameterDefaultNormalizedValue, 815
 - GetParameterIDFromIndex, 818
 - GetParameterIndex, 818
 - GetParameterIsAutomatable, 814
 - GetParameterName, 815
 - GetParameterNameOfLength, 815
 - GetParameterNormalizedValue, 821
 - GetParameterNumberOfSteps, 814
 - GetParameterOrientation, 817
 - GetParameterStringFromValue, 820
 - GetParameterType, 816
 - GetParameterValueFromString, 819
 - GetParameterValueInfo, 819
 - GetParameterValueString, 820
 - Initialize, 812
 - NotificationReceived, 812
 - ReleaseParameter, 823
 - ResetFieldData, 825
 - SetChunk, 828
 - SetCustomData, 830
 - SetParameterDefaultNormalizedValue, 816
 - SetParameterNormalizedRelative, 821
 - SetParameterNormalizedValue, 821
 - TimerWakeup, 829
 - TouchParameter, 822
 - Uninitialize, 812
 - UpdateParameterNormalizedRelative, 824
 - UpdateParameterNormalizedValue, 824
 - UpdateParameterTouch, 823
- AAX_IACFEffectParameters.h, 1531, 1532
- AAX_IACFEffectParameters_V2, 830
 - UpdateControlMIDINodes, 836
 - UpdateMIDINodes, 835
- AAX_IACFEffectParameters_V3, 836
- AAX_IACFEffectParameters_V4, 842
 - UpdatePageTable, 847
- AAX_IACFFeatureInfo, 848
 - AcquireProperties, 849
 - SupportLevel, 849
- AAX_IACFFeatureInfo.h, 1534
- AAX_IACFHostProcessor, 850
 - AnalyzeAudio, 855
 - Initialize, 852
 - InitOutputBounds, 852
 - PostAnalyze, 856
 - PostRender, 855
 - PreAnalyze, 856
 - PreRender, 854
 - RenderAudio, 854
 - SetLocation, 853
 - Uninitialize, 852
- AAX_IACFHostProcessor.h, 1535
- AAX_IACFHostProcessor_V2, 857
 - GetClipNameSuffix, 859
- AAX_IACFHostProcessorDelegate, 859
 - GetAudio, 860
 - GetSideChainInputNum, 861
- AAX_IACFHostProcessorDelegate.h, 1536, 1537
- AAX_IACFHostProcessorDelegate_V2, 862
 - ForceAnalyze, 863
- AAX_IACFHostProcessorDelegate_V3, 864
 - ForceProcess, 865
- AAX_IACFHostServices, 866
 - Assert, 867
 - Trace, 867
- AAX_IACFHostServices.h, 1537, 1538
- AAX_IACFHostServices_V2, 868
 - StackTrace, 869
- AAX_IACFHostServices_V3, 870

- HandleAssertFailure, 872
- AAX_IACFPageTable, 872
 - Clear, 874
 - ClearMappedParameter, 876
 - Empty, 874
 - GetMappedParameterID, 876
 - GetNumMappedParameterIDs, 876
 - GetNumPages, 875
 - InsertPage, 875
 - MapParameterID, 877
 - RemovePage, 875
- AAX_IACFPageTable.h, 1538, 1539
- AAX_IACFPageTable_V2, 878
 - ClearNameVariationsForParameter, 883
 - ClearParameterNameVariations, 883
 - GetNameVariationParameterIDAtIndex, 880
 - GetNumNameVariationsForParameter, 880
 - GetNumParametersWithNameVariations, 879
 - GetParameterNameVariationAtIndex, 881
 - GetParameterNameVariationOfLength, 882
 - SetParameterNameVariation, 884
- AAX_IACFPageTableController, 884
 - CopyTableForEffect, 886
 - CopyTableOfLayoutForEffect, 887
- AAX_IACFPageTableController.h, 1539, 1540
- AAX_IACFPageTableController_V2, 888
 - CopyTableForEffectFromFile, 889
 - CopyTableOfLayoutFromFile, 890
- AAX_IACFPrivateDataAccess, 891
 - ReadPortDirect, 892
 - WritePortDirect, 892
- AAX_IACFPrivateDataAccess.h, 1541
- AAX_IACFPropertyMap, 893
 - AddProperty, 895
 - GetProperty, 894
 - RemoveProperty, 895
- AAX_IACFPropertyMap.h, 1542
- AAX_IACFPropertyMap_V2, 895
 - AddPropertyWithIDArray, 897
 - GetPropertyWithIDArray, 897
- AAX_IACFPropertyMap_V3, 898
 - AddProperty64, 900
 - GetProperty64, 900
- AAX_IACFSessionDocument, 901
 - GetDocumentData, 902
- AAX_IACFSessionDocument.h, 1543, 1544
 - AAX_IACFSessionDocument_H, 1543
- AAX_IACFSessionDocument_H
 - AAX_IACFSessionDocument.h, 1543
- AAX_IACFSessionDocumentClient, 902
 - Initialize, 904
 - NotificationReceived, 905
 - SetSessionDocument, 904
 - Uninitialize, 904
- AAX_IACFSessionDocumentClient.h, 1544, 1545
 - AAX_IACFSessionDocumentClient_H, 1544
- AAX_IACFSessionDocumentClient_H
 - AAX_IACFSessionDocumentClient.h, 1544
- AAX_IACFTask, 905
 - AddResult, 908
 - GetArgumentOfType, 907
 - GetProgress, 908
 - GetType, 907
 - SetDone, 908
 - SetProgress, 907
- AAX_IACFTask.h, 1545, 1546
 - AAX_IACFTask_H, 1546
- AAX_IACFTask_H
 - AAX_IACFTask.h, 1546
- AAX_IACFTaskAgent, 909
 - AddTask, 911
 - CancelAllTasks, 911
 - Initialize, 910
 - Uninitialize, 911
- AAX_IACFTaskAgent.h, 1547
 - AAX_IACFTaskAgent_H, 1547
- AAX_IACFTaskAgent_H
 - AAX_IACFTaskAgent.h, 1547
- AAX_IACFTransport, 912
 - GetBarBeatPosition, 916
 - GetCurrentLoopPosition, 915
 - GetCurrentMeter, 914
 - GetCurrentNativeSampleLocation, 915
 - GetCurrentTempo, 913
 - GetCurrentTickPosition, 914
 - GetCurrentTicksPerBeat, 917
 - GetCustomTickPosition, 916
 - GetTicksPerQuarter, 916
 - IsTransportPlaying, 914
- AAX_IACFTransport.h, 1548, 1549
- AAX_IACFTransport_V2, 917
 - GetFeetFramesInfo, 920
 - GetTimeCodeInfo, 920
 - GetTimelineSelectionStartPosition, 919
 - IsMetronomeEnabled, 921
- AAX_IACFTransport_V3, 921
 - GetHDTTimeCodeInfo, 924
- AAX_IACFTransport_V4, 924
 - GetTimelineSelectionEndPosition, 927
- AAX_IACFTransportControl, 928
 - RequestTransportStart, 929
 - RequestTransportStop, 929
- AAX_IACFTransportControl.h, 1550
- AAX_IACFViewContainer, 930
 - GetModifiers, 932
 - GetPtr, 931
 - GetType, 931
 - HandleParameterMouseDown, 932
 - HandleParameterMouseDrag, 933
 - HandleParameterMouseUp, 933
 - SetViewSize, 932
- AAX_IACFViewContainer.h, 1551
- AAX_IACFViewContainer_V2, 934
 - HandleMultipleParametersMouseDown, 936
 - HandleMultipleParametersMouseDrag, 936
 - HandleMultipleParametersMouseUp, 937

- AAX_IACFViewContainer_V3, 937
 - HandleParameterMouseEnter, 940
 - HandleParameterMouseExit, 940
- AAX_IAutomationDelegate, 940
 - ~AAX_IAutomationDelegate, 941
 - GetTouchState, 945
 - ParameterNameChanged, 945
 - PostCurrentValue, 944
 - PostReleaseRequest, 944
 - PostSetValueRequest, 943
 - PostTouchRequest, 944
 - RegisterParameter, 942
 - UnregisterParameter, 942
- AAX_IAutomationDelegate.h, 1552
- AAX_ICollection, 946
 - ~AAX_ICollection, 947
 - AddEffect, 948
 - AddPackageName, 948
 - DescriptionHost, 950
 - HostDefinition, 950
 - NewDescriptor, 947
 - NewPropertyMap, 949
 - SetManufacturerName, 948
 - SetPackageVersion, 949
 - SetProperties, 949
- AAX_ICollection.h, 1553
- AAX_IComponentDescriptor, 951
 - ~AAX_IComponentDescriptor, 953
 - AddAudioBufferLength, 955
 - AddAudioIn, 953
 - AddAudioOut, 954
 - AddAuxOutputStem, 958
 - AddClock, 956
 - AddDataInPort, 957
 - AddDmaInstance, 960
 - AddMeters, 961
 - AddMIDINode, 962
 - AddPrivateData, 959
 - AddProcessProc, 966
 - AddProcessProc_Native, 964, 967
 - AddProcessProc_TI, 965
 - AddReservedField, 963
 - AddSampleRate, 955
 - AddSideChainIn, 957
 - AddTemporaryData, 960
 - Clear, 953
 - DuplicatePropertyMap, 964
 - NewPropertyMap, 963
- AAX_IComponentDescriptor.h, 1554, 1555
- AAX_IContainer, 968
 - ~AAX_IContainer, 969
 - Clear, 969
 - EStatus, 969
 - eStatus_NotInitialized, 969
 - eStatus_Overflow, 969
 - eStatus_Success, 969
 - eStatus_Unavailable, 969
 - eStatus_Unsupported, 969
- AAX_IContainer.h, 1556
- AAX_IController, 970
 - ~AAX_IController, 972
 - ClearMeterClipped, 981
 - ClearMeterPeakValue, 979
 - CreateTableCopyForEffect, 983
 - CreateTableCopyForEffectFromFile, 984
 - CreateTableCopyForLayout, 983
 - CreateTableCopyForLayoutFromFile, 985
 - GetCurrentAutomationTimestamp, 981
 - GetCurrentMeterValue, 978
 - GetCycleCount, 974
 - GetEffectID, 972
 - GetHostName, 982
 - GetInputStemFormat, 973
 - GetIsAudioSuite, 982
 - GetMeterClipped, 979
 - GetMeterCount, 979
 - GetMeterPeakValue, 978
 - GetNextMIDIAPacket, 981
 - GetOutputStemFormat, 973
 - GetPluginTargetPlatform, 982
 - GetSampleRate, 972
 - GetSignalLatency, 973
 - GetTODLocation, 974
 - PostPacket, 976
 - SendNotification, 977, 978
 - SetCycleCount, 976
 - SetSignalLatency, 975
- AAX_IController.h, 1557
- AAX_IDataBuffer, 986
 - AAX_DELETE, 987
 - AAX_OVERRIDE, 987
 - ACF_DECLARE_STANDARD_UNKNOWN, 987
- AAX_IDataBuffer.h, 1559, 1560
 - AAX_IDataBuffer_H, 1559
- AAX_IDataBuffer_H
 - AAX_IDataBuffer.h, 1559
- AAX_IDataBufferWrapper, 988
 - ~AAX_IDataBufferWrapper, 989
 - Data, 989
 - Size, 989
 - Type, 989
- AAX_IDataBufferWrapper.h, 1560, 1561
 - AAX_IDATABUFFERWRAPPER_H, 1561
- AAX_IDATABUFFERWRAPPER_H
 - AAX_IDataBufferWrapper.h, 1561
- AAX_IDescriptionHost, 990
 - ~AAX_IDescriptionHost, 990
 - AcquireFeatureProperties, 990
- AAX_IDescriptionHost.h, 1561, 1562
- AAX_IDisplayDelegate< T >, 991
 - Clone, 994
 - StringToValue, 995
 - ValueToString, 994, 995
- AAX_IDisplayDelegate.h, 1562
- AAX_IDisplayDelegateBase, 996
 - ~AAX_IDisplayDelegateBase, 997

- AAX_IDisplayDelegateDecorator
 - AAX_IDisplayDelegateDecorator< T >, 999
- AAX_IDisplayDelegateDecorator< T >, 997
 - ~AAX_IDisplayDelegateDecorator, 1000
 - AAX_IDisplayDelegateDecorator, 999
 - Clone, 1000
 - StringToValue, 1002
 - ValueToString, 1000, 1001
- AAX_IDisplayDelegateDecorator.h, 1563
- AAX_IDma, 1003
 - ~AAX_IDma, 1007
 - EMode, 1005
 - eMode_Burst, 1007
 - eMode_Error, 1007
 - eMode_Gather, 1007
 - eMode_Scatter, 1007
 - EState, 1005
 - eState_Complete, 1005
 - eState_Error, 1005
 - eState_Init, 1005
 - eState_Pending, 1005
 - eState_Running, 1005
 - GetBaseOffset, 1013
 - GetBurstLength, 1010
 - GetDmaMode, 1008
 - GetDmaState, 1008
 - GetDst, 1009
 - GetFifoBuffer, 1011
 - GetFifoSize, 1014
 - GetLinearBuffer, 1012
 - GetNumBursts, 1010
 - GetNumOffsets, 1013
 - GetOffsetTable, 1012
 - GetSrc, 1009
 - GetTransferSize, 1011
 - IsTransferComplete, 1007
 - PostRequest, 1007
 - SetBaseOffset, 1013
 - SetBurstLength, 1009
 - SetDmaState, 1008
 - SetDst, 1009
 - SetFifoBuffer, 1011
 - SetFifoSize, 1014
 - SetLinearBuffer, 1012
 - SetNumBursts, 1010
 - SetNumOffsets, 1013
 - SetOffsetTable, 1012
 - SetSrc, 1008
 - SetTransferSize, 1011
- AAX_IDma.h, 1565, 1566
 - AAX_DMA_API, 1565
 - AAX_IDMA_H, 1565
- AAX_IDMA_H
 - AAX_IDma.h, 1565
- AAX_IEffectDescriptor, 1014
 - ~AAX_IEffectDescriptor, 1015
 - AddCategory, 1017
 - AddCategoryBypassParameter, 1017
 - AddComponent, 1016
 - AddControlMIDINode, 1019
 - AddMeterDescription, 1019
 - AddName, 1017
 - AddProcPtr, 1018
 - AddResourceInfo, 1019
 - NewComponentDescriptor, 1016
 - NewPropertyMap, 1018
 - SetProperties, 1018
- AAX_IEffectDescriptor.h, 1567
- AAX_IEffectDirectData, 1020
 - AAX_DELETE, 1023
 - ACF_DECLARE_STANDARD_UNKNOWN, 1023
 - override, 1023
- AAX_IEffectDirectData.h, 1568
- AAX_IEffectGUI, 1024
 - AAX_DELETE, 1026
 - ACF_DECLARE_STANDARD_UNKNOWN, 1026
 - override, 1026
- AAX_IEffectGUI.h, 1569
- AAX_IEffectParameters, 1027
 - AAX_DELETE, 1033
 - ACF_DECLARE_STANDARD_UNKNOWN, 1033
 - override, 1033
- AAX_IEffectParameters.h, 1570
- AAX_IFeatureInfo, 1033
 - ~AAX_IFeatureInfo, 1034
 - AcquireProperties, 1034
 - ID, 1034
 - SupportLevel, 1034
- AAX_IFeatureInfo.h, 1570, 1571
- AAX_IHostProcessor, 1035
 - AAX_DELETE, 1037
 - ACF_DECLARE_STANDARD_UNKNOWN, 1037
 - override, 1038
- AAX_IHostProcessor.h, 1571, 1572
- AAX_IHostProcessorDelegate, 1038
 - ~AAX_IHostProcessorDelegate, 1039
 - ForceAnalyze, 1040
 - ForceProcess, 1040
 - GetAudio, 1039
 - GetSideChainInputNum, 1040
- AAX_IHostProcessorDelegate.h, 1572
- AAX_IHostServices, 1041
 - ~AAX_IHostServices, 1042
 - HandleAssertFailure, 1042
 - StackTrace, 1043
 - Trace, 1043
- AAX_IHostServices.h, 1573
- AAX_IMIDIMessageInfoDelegate, 1044
 - ~AAX_IMIDIMessageInfoDelegate, 1044
 - Accepts, 1045
 - Accepts_ExactStatus, 1046
 - Length, 1044
 - Mask, 1044
 - ToString, 1045
 - ToString_AppendByteRange, 1047
 - ToString_AppendCStr, 1046

- ToString_AppendNumber, [1046](#)
- ToString_AppendValid, [1047](#)
- AAX_IMIDINode, [1048](#)
 - ~AAX_IMIDINode, [1049](#)
 - GetNodeBuffer, [1049](#)
 - GetTransport, [1049](#)
 - PostMIDIPacket, [1049](#)
- AAX_IMIDINode.h, [1574](#)
- AAX_Init.h, [1575](#), [1579](#)
 - AAXCanUnloadNow, [1576](#)
 - AAXGetClassFactory, [1576](#)
 - AAXGetSDKVersion, [1578](#)
 - AAXRegisterComponent, [1575](#)
 - AAXShutdown, [1577](#)
 - AAXStartup, [1577](#)
- AAX_InstrumentParameters.doxygen, [1295](#)
- AAX_INT16_MAX
 - AAX_Enums.h, [1446](#)
- AAX_INT16_MIN
 - AAX_Enums.h, [1446](#)
- AAX_INT32_MAX
 - AAX_Enums.h, [1446](#)
- AAX_INT32_MIN
 - AAX_Enums.h, [1446](#)
- AAX_INT_HI
 - AAX_MiscUtils.h, [1717](#)
- AAX_INT_LO
 - AAX_MiscUtils.h, [1716](#)
- AAX_InterfaceList.doxygen, [1295](#)
- AAX_IPacketHandler, [1050](#)
 - ~AAX_IPacketHandler, [1051](#)
 - Call, [1051](#)
 - Clone, [1051](#)
- AAX_IPageTable, [1051](#)
 - ~AAX_IPageTable, [1052](#)
 - Clear, [1053](#)
 - ClearMappedParameter, [1056](#)
 - ClearNameVariationsForParameter, [1061](#)
 - ClearParameterNameVariations, [1061](#)
 - Empty, [1053](#)
 - GetMappedParameterID, [1056](#)
 - GetNameVariationParameterIDAtIndex, [1058](#)
 - GetNumMappedParameterIDs, [1055](#)
 - GetNumNameVariationsForParameter, [1059](#)
 - GetNumPages, [1053](#)
 - GetNumParametersWithNameVariations, [1057](#)
 - GetParameterNameVariationAtIndex, [1059](#)
 - GetParameterNameVariationOfLength, [1060](#)
 - InsertPage, [1055](#)
 - MapParameterID, [1057](#)
 - RemovePage, [1055](#)
 - SetParameterNameVariation, [1062](#)
- AAX_IPageTable.h, [1579](#)
- AAX_IParameter, [1063](#)
 - ~AAX_IParameter, [1066](#)
 - AddShortenedName, [1068](#)
 - Automatable, [1068](#)
 - ClearShortenedNames, [1068](#)
 - CloneValue, [1066](#)
 - GetBoolFromNormalizedValue, [1075](#)
 - GetDoubleFromNormalizedValue, [1077](#)
 - GetFloatFromNormalizedValue, [1076](#)
 - GetInt32FromNormalizedValue, [1076](#)
 - GetNormalizedDefaultValue, [1070](#)
 - GetNormalizedValue, [1070](#)
 - GetNormalizedValueFromBool, [1073](#)
 - GetNormalizedValueFromDouble, [1075](#)
 - GetNormalizedValueFromFloat, [1074](#)
 - GetNormalizedValueFromInt32, [1074](#)
 - GetNormalizedValueFromStep, [1072](#)
 - GetNormalizedValueFromString, [1075](#)
 - GetNumberOfSteps, [1071](#)
 - GetOrientation, [1084](#)
 - GetStepValue, [1071](#)
 - GetStepValueFromNormalizedValue, [1072](#)
 - GetStringFromNormalizedValue, [1077](#), [1078](#)
 - GetType, [1083](#)
 - GetValueAsBool, [1079](#)
 - GetValueAsDouble, [1080](#)
 - GetValueAsFloat, [1079](#)
 - GetValueAsInt32, [1079](#)
 - GetValueAsString, [1080](#)
 - GetValueString, [1072](#), [1073](#)
 - Identifier, [1066](#)
 - Name, [1067](#)
 - Release, [1070](#)
 - SetAutomationDelegate, [1069](#)
 - SetDisplayDelegate, [1084](#)
 - SetName, [1067](#)
 - SetNormalizedDefaultValue, [1070](#)
 - SetNormalizedValue, [1070](#)
 - SetNumberOfSteps, [1071](#)
 - SetOrientation, [1083](#)
 - SetStepValue, [1072](#)
 - SetTaperDelegate, [1084](#)
 - SetToDefaultValue, [1071](#)
 - SetType, [1083](#)
 - SetValueFromString, [1078](#)
 - SetValueWithBool, [1081](#)
 - SetValueWithDouble, [1082](#)
 - SetValueWithFloat, [1082](#)
 - SetValueWithInt32, [1081](#)
 - SetValueWithString, [1082](#)
 - ShortenedName, [1068](#)
 - Touch, [1069](#)
 - UpdateNormalizedValue, [1085](#)
- AAX_IParameter.h, [1580](#), [1581](#)
- AAX_IParameterValue, [1085](#)
 - ~AAX_IParameterValue, [1086](#)
 - Clone, [1086](#)
 - GetValueAsBool, [1087](#)
 - GetValueAsDouble, [1088](#)
 - GetValueAsFloat, [1088](#)
 - GetValueAsInt32, [1087](#)
 - GetValueAsString, [1088](#)
 - Identifier, [1086](#)

- AAX_IPointerQueue< T >, 1089
 - ~AAX_IPointerQueue, 1091
 - Clear, 1091
 - Peek, 1092
 - Pop, 1091
 - Push, 1091
 - template_type, 1090
 - value_type, 1090
- AAX_IPointerQueue.h, 1583
- AAX_IPrivateDataAccess, 1092
 - ~AAX_IPrivateDataAccess, 1093
 - ReadPortDirect, 1093
 - WritePortDirect, 1094
- AAX_IPrivateDataAccess.h, 1584
- AAX_IPropertyMap, 1094
 - ~AAX_IPropertyMap, 1096
 - AddPointerProperty, 1097, 1098
 - AddProperty, 1097
 - AddPropertyWithIDArray, 1099
 - GetIUnknown, 1099
 - GetPointerProperty, 1096
 - GetProperty, 1096
 - GetPropertyWithIDArray, 1099
 - RemoveProperty, 1098
- AAX_IPropertyMap.h, 1585
- AAX_ISessionDocument, 1100
 - ~AAX_ISessionDocument, 1101
 - GetDocumentData, 1101
 - GetTempoMap, 1101
 - Valid, 1101
- AAX_ISessionDocument.h, 1586
 - AAX_ISessionDocument_H, 1586
- AAX_ISessionDocument::TempoMap, 1288
 - ~TempoMap, 1289
 - Data, 1289
 - Size, 1289
- AAX_ISessionDocument_H
 - AAX_ISessionDocument.h, 1586
- AAX_ISessionDocumentClient, 1102
 - AAX_DELETE, 1104
 - ACF_DECLARE_STANDARD_UNKNOWN, 1104
 - override, 1104
- AAX_ISessionDocumentClient.h, 1587, 1588
 - AAX_ISessionDocumentClient_H, 1587
- AAX_ISessionDocumentClient_H
 - AAX_ISessionDocumentClient.h, 1587
- AAX_IString, 1104
 - ~AAX_IString, 1105
 - Get, 1106
 - Length, 1105
 - MaxLength, 1105
 - operator=, 1106
 - Set, 1106
- AAX_IString.h, 1588, 1589
- AAX_ITaperDelegate< T >, 1107
 - Clone, 1108
 - ConstrainRealValue, 1109
 - GetMaximumValue, 1109
 - GetMinimumValue, 1109
 - NormalizedToReal, 1110
 - RealToNormalized, 1110
- AAX_ITaperDelegate.h, 1589
- AAX_ITaperDelegateBase, 1111
 - ~AAX_ITaperDelegateBase, 1112
- AAX_ITask, 1113
 - ~AAX_ITask, 1113
 - AddResult, 1115
 - GetArgumentOfType, 1114
 - GetProgress, 1115
 - GetType, 1114
 - SetDone, 1115
 - SetProgress, 1114
- AAX_ITask.h, 1590, 1591
 - AAX_ITask_H, 1590
- AAX_ITask_H
 - AAX_ITask.h, 1590
- AAX_ITaskAgent, 1116
 - AAX_DELETE, 1118
 - AAX_OVERRIDE, 1118
 - ACF_DECLARE_STANDARD_UNKNOWN, 1118
- AAX_ITaskAgent.h, 1591
- AAX_ITransport, 1119
 - ~AAX_ITransport, 1120
 - GetBarBeatPosition, 1124
 - GetCurrentLoopPosition, 1122
 - GetCurrentMeter, 1121
 - GetCurrentNativeSampleLocation, 1123
 - GetCurrentTempo, 1120
 - GetCurrentTickPosition, 1122
 - GetCurrentTicksPerBeat, 1124
 - GetCustomTickPosition, 1123
 - GetFeetFramesInfo, 1125
 - GetHDTimeCodeInfo, 1126
 - GetTicksPerQuarter, 1124
 - GetTimeCodeInfo, 1125
 - GetTimelineSelectionEndPosition, 1127
 - GetTimelineSelectionStartPosition, 1125
 - IsMetronomeEnabled, 1126
 - IsTransportPlaying, 1121
 - RequestTransportStart, 1127
 - RequestTransportStop, 1127
- AAX_ITransport.h, 1592, 1593
- AAX_IViewContainer, 1128
 - ~AAX_IViewContainer, 1129
 - GetModifiers, 1130
 - GetPtr, 1130
 - GetType, 1130
 - HandleMultipleParametersMouseDown, 1133
 - HandleMultipleParametersMouseDrag, 1133
 - HandleMultipleParametersMouseUp, 1134
 - HandleParameterMouseDown, 1131
 - HandleParameterMouseDrag, 1131
 - HandleParameterMouseEnter, 1132
 - HandleParameterMouseExit, 1132
 - HandleParameterMouseUp, 1132
 - SetViewSize, 1130

- AAX_IViewContainer.h, [1593](#), [1594](#)
- AAX_LIMIT
 - AAX_SliderConversions.h, [1632](#)
- AAX_LinkedParameters.doxygen, [1295](#)
- AAX_LittleEndianNativeSwap
 - AAX_EndianSwap.h, [1433](#)
- AAX_LittleEndianNativeSwapInPlace
 - AAX_EndianSwap.h, [1432](#)
- AAX_LittleEndianNativeSwapSequenceInPlace
 - AAX_EndianSwap.h, [1435](#)
- AAX_LO
 - AAX_MiscUtils.h, [1716](#)
- AAX_Map, [1134](#)
 - ~AAX_Map, [1135](#)
 - AAX_Map, [1135](#)
 - GetCoefficient, [1135](#)
 - GetFirstX, [1136](#)
 - GetFirstY, [1136](#)
 - GetLastX, [1136](#)
 - GetLastY, [1136](#)
 - GetSize, [1136](#)
 - GetUpperBoundIndex, [1135](#)
 - GetX, [1136](#)
 - GetY, [1136](#)
 - SetCoefficients, [1135](#)
- AAX_Map.h, [1713](#), [1714](#)
 - AAX_MAP_H, [1713](#)
- AAX_MAP_H
 - AAX_Map.h, [1713](#)
- AAX_Media_Composer_Guide.doxygen, [1295](#)
- AAX_MIDILogging.cpp, [1296](#)
- AAX_MIDILogging.h, [1297](#)
- AAX_MIDIUtilities.h, [1595](#), [1596](#)
- AAX_MiscUtils.h, [1714](#), [1717](#)
 - AAX_ALIGNMENT_HINT, [1716](#)
 - AAX_DWORD_ALIGNED_HINT, [1716](#)
 - AAX_HI, [1716](#)
 - AAX_INT_HI, [1717](#)
 - AAX_INT_LO, [1716](#)
 - AAX_LO, [1716](#)
 - AAX_MISCUTILS_H, [1715](#)
 - AAX_WORD_ALIGNED_HINT, [1716](#)
- AAX_MISCUTILS_H
 - AAX_MiscUtils.h, [1715](#)
- AAX_OtherExtensions.doxygen, [1295](#)
- AAX_OVERRIDE
 - AAX.h, [1306](#)
 - AAX_IDataBuffer, [987](#)
 - AAX_ITaskAgent, [1118](#)
- AAX_Page_Table_Guide.doxygen, [1295](#)
- AAX_PageTableUtilities.h, [1598](#)
- AAX_ParameterAutomation.doxygen, [1295](#)
- AAX_ParameterManager.doxygen, [1295](#)
- AAX_ParameterUpdateProtocol.doxygen, [1295](#)
- AAX_ParameterUpdateTiming.doxygen, [1295](#)
- AAX_PlatformOptimizationConstants.h, [1720](#)
 - AAX_PLATFORMOPTIMIZATIONCONSTANTS_H, [1720](#)
- AAX_PLATFORMOPTIMIZATIONCONSTANTS_H
 - AAX_PlatformOptimizationConstants.h, [1720](#)
- AAX_PluginBundleLocation.h, [1297](#), [1298](#)
- AAX_Point, [1137](#)
 - AAX_GUITypes.h, [1515](#)
 - AAX_Point, [1137](#)
 - horz, [1138](#)
 - vert, [1138](#)
- AAX_PointerSize
 - AAX.h, [1309](#)
- AAX_PopStructAlignment.h, [1601](#)
- AAX_PostStructAlignmentHelper.h, [1602](#)
- AAX_PREPROCESSOR_CONCAT
 - AAX.h, [1311](#)
- AAX_PREPROCESSOR_CONCAT_HELPER
 - AAX.h, [1311](#)
- AAX_PreStructAlignmentHelper.h, [1602](#), [1603](#)
- AAX_Pro_Tools_Guide.doxygen, [1296](#)
- AAX_Properties.h, [1603](#), [1623](#)
 - AAX_ENUM_SIZE_CHECK, [1623](#)
 - AAX_EProperty, [1605](#)
 - AAX_eProperty_AllowPreviewWithoutAnalysis, [1615](#)
 - AAX_eProperty_AlwaysBypass, [1614](#)
 - AAX_eProperty_AudioBufferLength, [1611](#)
 - AAX_eProperty_AudiosuitePropsBase, [1614](#)
 - AAX_eProperty_CanBypass, [1613](#)
 - AAX_eProperty_Constraint_AlwaysProcess, [1622](#)
 - AAX_eProperty_Constraint_DoNotApplyDefaultSettings, [1622](#)
 - AAX_eProperty_Constraint_Location, [1619](#)
 - AAX_eProperty_Constraint_MultiMonoSupport, [1620](#)
 - AAX_eProperty_Constraint_NeverCache, [1620](#)
 - AAX_eProperty_Constraint_NeverUnload, [1619](#)
 - AAX_eProperty_Constraint_Topology, [1619](#)
 - AAX_eProperty_ConstraintBase, [1619](#)
 - AAX_eProperty_ConstraintBase_2, [1621](#)
 - AAX_eProperty_ContinuousOnly, [1616](#)
 - AAX_eProperty_DebugPropertiesBase, [1622](#)
 - AAX_eProperty_Deprecated_DSP_Plugin_List, [1609](#)
 - AAX_eProperty_Deprecated_Native_Plugin_List, [1610](#)
 - AAX_eProperty_Deprecated_Plugin_List, [1609](#)
 - AAX_eProperty_DestinationTrack, [1616](#)
 - AAX_eProperty_DisableAudiosuiteReverse, [1618](#)
 - AAX_eProperty_DisableHandles, [1617](#)
 - AAX_eProperty_DisablePreview, [1617](#)
 - AAX_eProperty_DoesntIncrOutputSample, [1617](#)
 - AAX_eProperty_DSP_AudioBufferLength, [1611](#)
 - AAX_eProperty_EnableHostDebugLogs, [1623](#)
 - AAX_eProperty_ExternalProcessorTypeID, [1610](#)
 - AAX_eProperty_FeaturesBase, [1620](#)
 - AAX_eProperty_GeneralPropsBase, [1611](#)
 - AAX_eProperty_GUIBase, [1618](#)
 - AAX_eProperty_HybridInputStemFormat, [1614](#)
 - AAX_eProperty_HybridOutputStemFormat, [1614](#)

- AAX_eProperty_InputStemFormat, 1611
- AAX_eProperty_LatencyContribution, 1612
- AAX_eProperty_ManufacturerID, 1606
- AAX_eProperty_MaxASProp, 1618
- AAX_eProperty_MaxCap, 1623
- AAX_eProperty_MaxConstraintProp, 1620
- AAX_eProperty_MaxConstraintProp_2, 1622
- AAX_eProperty_MaxFeaturesProp, 1621
- AAX_eProperty_MaxGUIProp, 1618
- AAX_eProperty_MaxMeterProp, 1619
- AAX_eProperty_MaxProp, 1623
- AAX_eProperty_Meter_Ballistics, 1619
- AAX_eProperty_Meter_Orientation, 1619
- AAX_eProperty_Meter_Type, 1619
- AAX_eProperty_MeterBase, 1618
- AAX_eProperty_MinProp, 1606
- AAX_eProperty_MultiInputModeOnly, 1616
- AAX_eProperty_NativeBackgroundProc, 1611
- AAX_eProperty_NativeInstanceInitProc, 1610
- AAX_eProperty_NativeProcessProc, 1610
- AAX_eProperty_NeedsOutputDithered, 1618
- AAX_eProperty_NoID, 1606
- AAX_eProperty_NumberOfInputs, 1617
- AAX_eProperty_NumberOfOutputs, 1617
- AAX_eProperty_ObservesTransportState, 1621
- AAX_eProperty_OptionalAnalysis, 1615
- AAX_eProperty_OutputStemFormat, 1611
- AAX_eProperty_PluginID_AudioSuite, 1607
- AAX_eProperty_PluginID_Deprecated, 1609
- AAX_eProperty_PluginID_ExternalProcessor, 1610
- AAX_eProperty_PluginID_Native, 1607
- AAX_eProperty_PluginID_NoProcessing, 1608
- AAX_eProperty_PluginID_RTAS, 1607
- AAX_eProperty_PluginID_TI, 1608
- AAX_eProperty_PluginSpecPropsBase, 1606
- AAX_eProperty_ProcessProcPropsBase, 1610
- AAX_eProperty_ProductID, 1606
- AAX_eProperty_Related_DSP_Plugin_List, 1609
- AAX_eProperty_Related_Native_Plugin_List, 1609
- AAX_eProperty_RequestsAllTrackData, 1616
- AAX_eProperty_RequiresAnalysis, 1615
- AAX_eProperty_RequiresChunkCallsOnMainThread, 1621
- AAX_eProperty_SampleRate, 1612
- AAX_eProperty_ShowInMenus, 1614
- AAX_eProperty_SideChainStemFormat, 1613
- AAX_eProperty_StoreXMLPageTablesByEffect, 1621
- AAX_eProperty_StoreXMLPageTablesByType, 1621
- AAX_eProperty_SupportsSaveRestore, 1620
- AAX_eProperty_SupportsSideChainInput, 1618
- AAX_eProperty_TI_ForceAllowChipSharing, 1614
- AAX_eProperty_TI_InstanceCycleCount, 1613
- AAX_eProperty_TI_MaxInstancesPerChip, 1613
- AAX_eProperty_TI_SharedCycleCount, 1613
- AAX_eProperty_TIBackgroundProc, 1611
- AAX_eProperty_TIDLLFileName, 1611
- AAX_eProperty_TIInstanceInitProc, 1611
- AAX_eProperty_TIProcessProc, 1611
- AAX_eProperty_UsesClientGUI, 1618
- AAX_eProperty_UsesRandomAccess, 1615
- AAX_eProperty_UsesTransport, 1620
- AAX_eProperty_UsesTransportControl, 1621
- AAX_Push2ByteStructAlignment.h, 1626
- AAX_Push4ByteStructAlignment.h, 1627, 1628
- AAX_Push8ByteStructAlignment.h, 1628, 1629
- AAX_Quantize.h, 1721, 1722
 - AAX_QUANTIZE_H, 1721
- AAX_QUANTIZE_H
 - AAX_Quantize.h, 1721
- AAX_RandomGen.h, 1723, 1724
 - AAX_RANDOMGEN_H, 1724
- AAX_RANDOMGEN_H
 - AAX_RandomGen.h, 1724
- AAX_RealTimePerformance.doxygen, 1296
- AAX_Rect, 1138
 - AAX_GUITypes.h, 1515
 - AAX_Rect, 1138, 1139
 - height, 1139
 - left, 1139
 - top, 1139
 - width, 1139
- AAX_RelatedTypes.doxygen, 1296
- AAX_Result
 - AAX.h, 1313
- AAX_RESULT_ADD_FIELD_UNSUPPORTED_FIELD_TYPE
 - AAX_Errors.h, 1492
- AAX_RESULT_NEW_PACKET_POSTED
 - AAX_Errors.h, 1492
- AAX_RESULT_PACKET_STREAM_NOT_EMPTY
 - AAX_Errors.h, 1492
- AAX_SampleRateUtils.h, 1725, 1728
 - CoarseSampleRate, 1726
 - CoarseSampleRateFactor, 1727
 - CoarseSampleRateIndex, 1727
 - ESRUtils, 1726
 - eSRUtils_192kIndex, 1726
 - eSRUtils_192kRangeCoarse, 1726
 - eSRUtils_192kRangeMax, 1726
 - eSRUtils_192kRangeMin, 1726
 - eSRUtils_48kIndex, 1726
 - eSRUtils_48kRangeCoarse, 1726
 - eSRUtils_48kRangeMax, 1726
 - eSRUtils_48kRangeMin, 1726
 - eSRUtils_96kIndex, 1726
 - eSRUtils_96kRangeCoarse, 1726
 - eSRUtils_96kRangeMax, 1726
 - eSRUtils_96kRangeMin, 1726
- AAX_SCOPE_COMPUTE_DENORMALS
 - AAX_Denormal.h, 1704
- AAX_SCOPE_DENORMALS_AS_ZERO
 - AAX_Denormal.h, 1704
- AAX_SDK_1p0p1_REVISION
 - AAX_Version.h, 1684

- AAX_SDK_1p0p2_REVISION
 - AAX_Version.h, [1684](#)
- AAX_SDK_1p0p3_REVISION
 - AAX_Version.h, [1684](#)
- AAX_SDK_1p0p4_REVISION
 - AAX_Version.h, [1684](#)
- AAX_SDK_1p0p5_REVISION
 - AAX_Version.h, [1684](#)
- AAX_SDK_1p0p6_REVISION
 - AAX_Version.h, [1684](#)
- AAX_SDK_1p5p0_REVISION
 - AAX_Version.h, [1684](#)
- AAX_SDK_2p0b1_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p0p0_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p0p1_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p1p0_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p1p1_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p2p0_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p2p1_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p2p2_REVISION
 - AAX_Version.h, [1685](#)
- AAX_SDK_2p3p0_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_2p3p1_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_2p3p2_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_2p4p0_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_2p4p1_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_2p5p0_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_2p6p0_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_2p6p1_REVISION
 - AAX_Version.h, [1686](#)
- AAX_SDK_ChangeLog.doxygen, [1296](#)
- AAX_SDK_CURRENT_REVISION
 - AAX_Version.h, [1683](#)
- AAX_SDK_ExamplePlugIns.doxygen, [1296](#)
- AAX_SDK_GUIExtensions.doxygen, [1296](#)
- AAX_SDK_VERSION
 - AAX_Version.h, [1683](#)
- AAX_SessionDocumentTypes.h, [1630](#), [1631](#)
 - AAX_SessionDocumentTypes_H, [1630](#)
 - kAAX_DataBufferType_TempoBreakpointArray, [1630](#)
- AAX_SessionDocumentTypes_H
 - AAX_SessionDocumentTypes.h, [1630](#)
- AAX_SHybridRenderInfo, [1140](#)
 - mAudioInputs, [1140](#)
 - mAudioOutputs, [1140](#)
 - mClock, [1141](#)
 - mNumAudioInputs, [1140](#)
 - mNumAudioOutputs, [1141](#)
 - mNumSamples, [1141](#)
- AAX_SInstrumentPrivateData, [1141](#)
 - mMonolithicParametersPtr, [1142](#)
- AAX_SInstrumentRenderInfo, [1142](#)
 - mAdditionalInputMIDINodes, [1144](#)
 - mAudioInputs, [1143](#)
 - mAudioOutputs, [1143](#)
 - mClock, [1144](#)
 - mCurrentStateNum, [1145](#)
 - mGlobalNode, [1144](#)
 - mInputNode, [1144](#)
 - mMeters, [1145](#)
 - mNumSamples, [1143](#)
 - mPrivateData, [1144](#)
 - mTransportNode, [1144](#)
- AAX_SInstrumentSetupInfo, [1145](#)
 - AAX_SInstrumentSetupInfo, [1147](#)
 - mAudiosuiteID, [1153](#)
 - mAuxOutputStemFormats, [1150](#)
 - mAuxOutputStemNames, [1150](#)
 - mCanBypass, [1152](#)
 - mGlobalMIDIEventMask, [1147](#)
 - mGlobalMIDINodeName, [1147](#)
 - mHybridInputStemFormat, [1150](#)
 - mHybridOutputStemFormat, [1151](#)
 - mInputMIDIChannelMask, [1148](#)
 - mInputMIDINodeName, [1148](#)
 - mInputStemFormat, [1151](#)
 - mManufacturerID, [1152](#)
 - mMeterIDs, [1149](#)
 - mMultiMonoSupport, [1153](#)
 - mNeedsGlobalMIDI, [1147](#)
 - mNeedsInputMIDI, [1148](#)
 - mNeedsTransport, [1149](#)
 - mNumAdditionalInputMIDINodes, [1148](#)
 - mNumAuxOutputStems, [1150](#)
 - mNumMeters, [1149](#)
 - mOutputStemFormat, [1151](#)
 - mPluginID, [1152](#)
 - mProductID, [1152](#)
 - mTransportMIDINodeName, [1149](#)
 - mUseHostGeneratedGUI, [1151](#)
- AAX_SliderConversions.h, [1631](#), [1635](#)
 - AAX_LIMIT, [1632](#)
 - AAX_SLIDERCONVERSIONS_H, [1632](#)
 - DoubleToLongControl, [1633](#)
 - DoubleToLongControlNonlinear, [1633](#)
 - LogDoubleToLongControl, [1634](#)
 - LongControlToDouble, [1633](#)
 - LongControlToDoubleNonlinear, [1633](#)
 - LongControlToLogDouble, [1634](#)
 - LongControlToNewRange, [1632](#)
 - LongToLongControl, [1633](#)

- AAX_SLIDERCONVERSIONS_H
 - AAX_SliderConversions.h, [1632](#)
- AAX_SPlugInChunk, [1153](#)
 - AAX.h, [1317](#)
 - fChunkID, [1155](#)
 - fData, [1155](#)
 - fManufacturerID, [1154](#)
 - fName, [1155](#)
 - fPlugInID, [1155](#)
 - fProductID, [1154](#)
 - fSize, [1154](#)
 - fVersion, [1154](#)
- AAX_SPlugInChunkHeader, [1156](#)
 - AAX.h, [1316](#)
 - fChunkID, [1157](#)
 - fManufacturerID, [1157](#)
 - fName, [1158](#)
 - fPlugInID, [1157](#)
 - fProductID, [1157](#)
 - fSize, [1157](#)
 - fVersion, [1157](#)
- AAX_SPlugInChunkPtr
 - AAX.h, [1317](#)
- AAX_SPlugInIdentifierTriad, [1158](#)
 - AAX.h, [1317](#)
 - mManufacturerID, [1158](#)
 - mPlugInID, [1159](#)
 - mProductID, [1159](#)
- AAX_SPlugInIdentifierTriadPtr
 - AAX.h, [1317](#)
- AAX_STACKTRACE
 - AAX_Assert.h, [1328](#)
- AAX_STACKTRACE_RELEASE
 - AAX_Assert.h, [1326](#)
- AAX_STEM_FORMAT
 - AAX_Enums.h, [1447](#)
- AAX_STEM_FORMAT_CHANNEL_COUNT
 - AAX_Enums.h, [1447](#)
- AAX_STEM_FORMAT_INDEX
 - AAX_Enums.h, [1447](#)
- AAX_StLock_Guard, [1159](#)
 - ~AAX_StLock_Guard, [1160](#)
 - AAX_StLock_Guard, [1160](#)
- AAX_StringUtilities.h, [1635](#), [1636](#)
 - AAXLibrary_AAX_StringUtilities_h, [1636](#)
- AAX_StringUtilities.hpp, [1637](#), [1638](#)
 - DEFINE_AAX_ERROR_STRING, [1638](#)
- AAX_SUCCESS
 - AAX_Errors.h, [1492](#)
- AAX_SWALLOW
 - AAX_Exception.h, [1504](#)
- AAX_SWALLOW_MULT
 - AAX_Exception.h, [1504](#)
- AAX_TaskCompletionStatus
 - Task agent interface, [104](#)
- AAX_TI_Guide.doxygen, [1296](#)
- AAX_TRACE
 - AAX_Assert.h, [1328](#)
- AAX_TRACE_RELEASE
 - AAX_Assert.h, [1326](#)
- AAX_TRACEORSTACKTRACE
 - AAX_Assert.h, [1328](#)
- AAX_TRACEORSTACKTRACE_RELEASE
 - AAX_Assert.h, [1326](#)
- AAX_TransportStateInfo_V1, [1161](#)
 - AAX_TransportStateInfo_V1, [1161](#)
 - mIsLoopEnabled, [1163](#)
 - mIsRecordEnabled, [1162](#)
 - mIsRecording, [1163](#)
 - mRecordMode, [1162](#)
 - mTransportState, [1162](#)
 - ToString, [1162](#)
- AAX_TransportTypes.h, [1649](#), [1650](#)
 - operator!=, [1649](#)
 - operator==, [1649](#)
- AAX_Troubleshooting.doxygen, [1296](#)
- AAX_UIDs.h, [1651](#), [1668](#)
 - AAX_CompID_DescriptionHost, [1662](#)
 - AAX_CompID_FeatureInfo, [1663](#)
 - AAX_DocumentData_UID, [1654](#)
 - AAX_DocumentDataType_TempoMap, [1668](#)
 - AAX_Feature_UID, [1654](#)
 - AAXATTR_Client_Level, [1668](#)
 - AAXATTR_ClientFeature_AuxOutputStem, [1667](#)
 - AAXATTR_ClientFeature_MIDI, [1667](#)
 - AAXATTR_ClientFeature_SideChainInput, [1667](#)
 - AAXATTR_ClientFeature_StemFormat, [1667](#)
 - AAXCompID_AAXCollection, [1655](#)
 - AAXCompID_AAXComponentDescriptor, [1656](#)
 - AAXCompID_AAXEffectDescriptor, [1656](#)
 - AAXCompID_AAXPropertyMap, [1657](#)
 - AAXCompID_AutomationDelegate, [1658](#)
 - AAXCompID_Controller, [1658](#)
 - AAXCompID_DataBuffer, [1666](#)
 - AAXCompID_EffectDirectData, [1665](#)
 - AAXCompID_EffectGUI, [1665](#)
 - AAXCompID_EffectParameters, [1664](#)
 - AAXCompID_HostProcessor, [1664](#)
 - AAXCompID_HostProcessorDelegate, [1657](#)
 - AAXCompID_HostServices, [1655](#)
 - AAXCompID_PageTable, [1662](#)
 - AAXCompID_PageTableController, [1659](#)
 - AAXCompID_PrivateDataAccess, [1660](#)
 - AAXCompID_SessionDocument, [1663](#)
 - AAXCompID_SessionDocumentClient, [1666](#)
 - AAXCompID_Task, [1663](#)
 - AAXCompID_TaskAgent, [1666](#)
 - AAXCompID_Transport, [1661](#)
 - AAXCompID_TransportControl, [1661](#)
 - AAXCompID_ViewContainer, [1660](#)
 - IID_IAAXAutomationDelegateV1, [1658](#)
 - IID_IAAXCollectionV1, [1655](#)
 - IID_IAAXComponentDescriptorV1, [1656](#)
 - IID_IAAXComponentDescriptorV2, [1656](#)
 - IID_IAAXComponentDescriptorV3, [1657](#)
 - IID_IAAXControllerV1, [1659](#)

- IID_IAAXControllerV2, [1659](#)
- IID_IAAXControllerV3, [1659](#)
- IID_IAAXDataBufferV1, [1667](#)
- IID_IAAXDescriptionHostV1, [1662](#)
- IID_IAAXEffectDescriptorV1, [1656](#)
- IID_IAAXEffectDescriptorV2, [1656](#)
- IID_IAAXEffectDirectDataV1, [1665](#)
- IID_IAAXEffectDirectDataV2, [1666](#)
- IID_IAAXEffectGUIV1, [1665](#)
- IID_IAAXEffectParametersV1, [1664](#)
- IID_IAAXEffectParametersV2, [1664](#)
- IID_IAAXEffectParametersV3, [1664](#)
- IID_IAAXEffectParametersV4, [1664](#)
- IID_IAAXFeatureInfoV1, [1663](#)
- IID_IAAXHostProcessorDelegateV1, [1658](#)
- IID_IAAXHostProcessorDelegateV2, [1658](#)
- IID_IAAXHostProcessorDelegateV3, [1658](#)
- IID_IAAXHostProcessorV1, [1665](#)
- IID_IAAXHostProcessorV2, [1665](#)
- IID_IAAXHostServicesV1, [1655](#)
- IID_IAAXHostServicesV2, [1655](#)
- IID_IAAXHostServicesV3, [1655](#)
- IID_IAAXPageTableController, [1659](#)
- IID_IAAXPageTableControllerV2, [1659](#)
- IID_IAAXPageTableV1, [1662](#)
- IID_IAAXPageTableV2, [1662](#)
- IID_IAAXPrivateDataAccessV1, [1660](#)
- IID_IAAXPropertyMapV1, [1657](#)
- IID_IAAXPropertyMapV2, [1657](#)
- IID_IAAXPropertyMapV3, [1657](#)
- IID_IAAXSessionDocumentClientV1, [1666](#)
- IID_IAAXSessionDocumentV1, [1663](#)
- IID_IAAXTaskAgentV1, [1666](#)
- IID_IAAXTaskV1, [1663](#)
- IID_IAAXTransportControlV1, [1662](#)
- IID_IAAXTransportV1, [1661](#)
- IID_IAAXTransportV2, [1661](#)
- IID_IAAXTransportV3, [1661](#)
- IID_IAAXTransportV4, [1661](#)
- IID_IAAXViewContainerV1, [1660](#)
- IID_IAAXViewContainerV2, [1660](#)
- IID_IAAXViewContainerV3, [1660](#)
- AAX_UINT16_MAX
 - AAX_Enums.h, [1446](#)
- AAX_UINT16_MIN
 - AAX_Enums.h, [1446](#)
- AAX_UINT32_MAX
 - AAX_Enums.h, [1446](#)
- AAX_UINT32_MIN
 - AAX_Enums.h, [1446](#)
- AAX_UNIQUE_PTR
 - AAX.h, [1308](#)
- AAX_UtilsNative.h, [1671](#), [1672](#)
 - _AAX_UTILSNATIVE_H_, [1672](#)
- AAX_VAutomationDelegate, [1163](#)
 - ~AAX_VAutomationDelegate, [1165](#)
 - AAX_VAutomationDelegate, [1164](#)
 - GetTouchState, [1167](#)
 - GetUnknown, [1165](#)
 - ParameterNameChanged, [1168](#)
 - PostCurrentValue, [1166](#)
 - PostReleaseRequest, [1167](#)
 - PostSetValueRequest, [1166](#)
 - PostTouchRequest, [1167](#)
 - RegisterParameter, [1165](#)
 - UnregisterParameter, [1165](#)
- AAX_VAutomationDelegate.h, [1673](#)
- AAX_VCollection, [1168](#)
 - ~AAX_VCollection, [1170](#)
 - AAX_VCollection, [1170](#)
 - AddEffect, [1170](#)
 - AddPackageName, [1171](#)
 - DescriptionHost, [1173](#), [1174](#)
 - GetUnknown, [1174](#)
 - HostDefinition, [1174](#)
 - NewDescriptor, [1170](#)
 - NewPropertyMap, [1173](#)
 - SetManufacturerName, [1171](#)
 - SetPackageVersion, [1173](#)
 - SetProperties, [1173](#)
- AAX_VCollection.h, [1674](#)
- AAX_VComponentDescriptor, [1175](#)
 - ~AAX_VComponentDescriptor, [1178](#)
 - AAX_VComponentDescriptor, [1178](#)
 - AAX_VPropertyMap, [1190](#)
 - AddAudioBufferLength, [1180](#)
 - AddAudioIn, [1179](#)
 - AddAudioOut, [1179](#)
 - AddAuxOutputStem, [1182](#)
 - AddClock, [1180](#)
 - AddDataInPort, [1181](#)
 - AddDmaInstance, [1184](#)
 - AddMeters, [1184](#)
 - AddMIDINode, [1186](#)
 - AddPrivateData, [1183](#)
 - AddProcessProc, [1188](#)
 - AddProcessProc_Native, [1187](#)
 - AddProcessProc_TI, [1188](#)
 - AddReservedField, [1178](#)
 - AddSampleRate, [1180](#)
 - AddSideChainIn, [1181](#)
 - AddTemporaryData, [1183](#)
 - Clear, [1178](#)
 - DuplicatePropertyMap, [1187](#)
 - GetUnknown, [1190](#)
 - NewPropertyMap, [1186](#)
- AAX_VComponentDescriptor.h, [1675](#), [1676](#)
- AAX_VController, [1190](#)
 - ~AAX_VController, [1192](#)
 - AAX_VController, [1192](#)
 - ClearMeterClipped, [1202](#)
 - ClearMeterPeakValue, [1201](#)
 - CreateTableCopyForEffect, [1204](#)
 - CreateTableCopyForEffectFromFile, [1205](#)
 - CreateTableCopyForLayout, [1205](#)
 - CreateTableCopyForLayoutFromFile, [1206](#)

- GetCurrentAutomationTimestamp, 1196
- GetCurrentMeterValue, 1201
- GetCycleCount, 1195
- GetEffectID, 1193
- GetHostName, 1197
- GetHybridSignalLatency, 1194
- GetInputStemFormat, 1193
- GetIsAudioSuite, 1195
- GetMeterClipped, 1202
- GetMeterCount, 1202
- GetMeterPeakValue, 1201
- GetNextMIDIPacket, 1204
- GetOutputStemFormat, 1193
- GetPluginTargetPlatform, 1195
- GetSampleRate, 1193
- GetSignalLatency, 1194
- GetTODLocation, 1196
- PostPacket, 1199
- SendNotification, 1200
- SetCycleCount, 1198
- SetSignalLatency, 1197
- AAX_VController.h, 1677
- AAX_VDataBufferWrapper, 1207
 - ~AAX_VDataBufferWrapper, 1208
 - AAX_VDataBufferWrapper, 1208
 - Data, 1209
 - Size, 1209
 - Type, 1208
- AAX_VDataBufferWrapper.h, 1679, 1680
 - AAX_VDATABUFFERWRAPPER_H, 1679
- AAX_VDATABUFFERWRAPPER_H
 - AAX_VDataBufferWrapper.h, 1679
- AAX_VDescriptionHost, 1210
 - ~AAX_VDescriptionHost, 1211
 - AAX_VDescriptionHost, 1211
 - AcquireFeatureProperties, 1211
 - DescriptionHost, 1212
 - HostDefinition, 1212
 - Supported, 1212
- AAX_VDescriptionHost.h, 1680
- AAX_VEffectDescriptor, 1213
 - ~AAX_VEffectDescriptor, 1215
 - AAX_VEffectDescriptor, 1214
 - AddCategory, 1216
 - AddCategoryBypassParameter, 1216
 - AddComponent, 1215
 - AddControlMIDINode, 1218
 - AddMeterDescription, 1218
 - AddName, 1215
 - AddProcPtr, 1217
 - AddResourceInfo, 1217
 - GetUnknown, 1219
 - NewComponentDescriptor, 1215
 - NewPropertyMap, 1217
 - SetProperties, 1217
- AAX_VEffectDescriptor.h, 1681, 1682
- AAX_VENUE_Guide.doxygen, 1296
- AAX_Version.h, 1682, 1687
 - _AAX_VERSION_H, 1683
 - AAX_SDK_1p0p1_REVISION, 1684
 - AAX_SDK_1p0p2_REVISION, 1684
 - AAX_SDK_1p0p3_REVISION, 1684
 - AAX_SDK_1p0p4_REVISION, 1684
 - AAX_SDK_1p0p5_REVISION, 1684
 - AAX_SDK_1p0p6_REVISION, 1684
 - AAX_SDK_1p5p0_REVISION, 1684
 - AAX_SDK_2p0b1_REVISION, 1685
 - AAX_SDK_2p0p0_REVISION, 1685
 - AAX_SDK_2p0p1_REVISION, 1685
 - AAX_SDK_2p1p0_REVISION, 1685
 - AAX_SDK_2p1p1_REVISION, 1685
 - AAX_SDK_2p2p0_REVISION, 1685
 - AAX_SDK_2p2p1_REVISION, 1685
 - AAX_SDK_2p2p2_REVISION, 1685
 - AAX_SDK_2p3p0_REVISION, 1686
 - AAX_SDK_2p3p1_REVISION, 1686
 - AAX_SDK_2p3p2_REVISION, 1686
 - AAX_SDK_2p4p0_REVISION, 1686
 - AAX_SDK_2p4p1_REVISION, 1686
 - AAX_SDK_2p5p0_REVISION, 1686
 - AAX_SDK_2p6p0_REVISION, 1686
 - AAX_SDK_2p6p1_REVISION, 1686
 - AAX_SDK_CURRENT_REVISION, 1683
 - AAX_SDK_VERSION, 1683
- AAX_VFeatureInfo, 1219
 - ~AAX_VFeatureInfo, 1220
 - AAX_VFeatureInfo, 1220
 - AcquireProperties, 1221
 - ID, 1221
 - SupportLevel, 1220
- AAX_VFeatureInfo.h, 1687, 1688
- AAX_VHostProcessorDelegate, 1222
 - AAX_VHostProcessorDelegate, 1223
 - ForceAnalyze, 1224
 - ForceProcess, 1224
 - GetAudio, 1223
 - GetSideChainInputNum, 1224
- AAX_VHostProcessorDelegate.h, 1688, 1689
- AAX_VHostServices, 1225
 - ~AAX_VHostServices, 1226
 - AAX_VHostServices, 1226
 - HandleAssertFailure, 1226
 - StackTrace, 1227
 - Trace, 1227
- AAX_VHostServices.h, 1689, 1690
- AAX_VPageTable, 1228
 - ~AAX_VPageTable, 1231
 - AAX_VPageTable, 1231
 - AsUnknown, 1240
 - Clear, 1231
 - ClearMappedParameter, 1233
 - ClearNameVariationsForParameter, 1238
 - ClearParameterNameVariations, 1238
 - Empty, 1231
 - GetMappedParameterID, 1234
 - GetNameVariationParameterIDAtIndex, 1235

- GetNumMappedParameterIDs, 1233
- GetNumNameVariationsForParameter, 1236
- GetNumPages, 1232
- GetNumParametersWithNameVariations, 1235
- GetParameterNameVariationAtIndex, 1236
- GetParameterNameVariationOfLength, 1237
- InsertPage, 1232
- IsSupported, 1240
- MapParameterID, 1234
- RemovePage, 1232
- SetParameterNameVariation, 1239
- AAX_VPageTable.h, 1690, 1691
- AAX_VPrivateDataAccess, 1241
 - ~AAX_VPrivateDataAccess, 1242
 - AAX_VPrivateDataAccess, 1242
 - ReadPortDirect, 1242
 - WritePortDirect, 1243
- AAX_VPrivateDataAccess.h, 1692
- AAX_VPropertyMap, 1243
 - ~AAX_VPropertyMap, 1245
 - AAX_VComponentDescriptor, 1190
 - Acquire, 1246
 - AddPointerProperty, 1247, 1248
 - AddProperty, 1247
 - AddPropertyWithIDArray, 1248
 - Create, 1245
 - GetUnknown, 1249
 - GetPointerProperty, 1246
 - GetProperty, 1246
 - GetPropertyWithIDArray, 1249
 - RemoveProperty, 1248
- AAX_VPropertyMap.h, 1693
- AAX_VSessionDocument, 1250
 - ~AAX_VSessionDocument, 1251
 - AAX_VSessionDocument, 1251
 - Clear, 1251
 - GetDocumentData, 1252
 - GetTempoMap, 1251
 - Valid, 1251
- AAX_VSessionDocument.h, 1694, 1695
 - AAX_VSessionDocument_H, 1694
- AAX_VSessionDocument::VTempoMap, 1290
 - ~VTempoMap, 1291
 - Data, 1291
 - Size, 1291
 - VTempoMap, 1291
- AAX_VSessionDocument_H
 - AAX_VSessionDocument.h, 1694
- AAX_VTask, 1252
 - ~AAX_VTask, 1254
 - AAX_VTask, 1254
 - AddResult, 1255
 - GetArgumentOfType, 1254
 - GetProgress, 1255
 - GetType, 1254
 - SetDone, 1256
 - SetProgress, 1255
- AAX_VTask.h, 1695, 1696
 - AAX_VTask_H, 1696
- AAX_VTask_H
 - AAX_VTask.h, 1696
- AAX_VTransport, 1256
 - ~AAX_VTransport, 1259
 - AAX_VTransport, 1259
 - GetBarBeatPosition, 1262
 - GetCurrentLoopPosition, 1261
 - GetCurrentMeter, 1260
 - GetCurrentNativeSampleLocation, 1261
 - GetCurrentTempo, 1259
 - GetCurrentTickPosition, 1260
 - GetCurrentTicksPerBeat, 1263
 - GetCustomTickPosition, 1262
 - GetFeetFramesInfo, 1264
 - GetHDTIMECodeInfo, 1265
 - GetTicksPerQuarter, 1263
 - GetTimeCodeInfo, 1264
 - GetTimelineSelectionEndPosition, 1265
 - GetTimelineSelectionStartPosition, 1263
 - IsMetronomeEnabled, 1264
 - IsTransportPlaying, 1260
 - RequestTransportStart, 1266
 - RequestTransportStop, 1266
- AAX_VTransport.h, 1696, 1697
- AAX_VViewContainer, 1267
 - ~AAX_VViewContainer, 1268
 - AAX_VViewContainer, 1268
 - GetModifiers, 1269
 - GetPtr, 1269
 - GetType, 1268
 - HandleMultipleParametersMouseDown, 1272
 - HandleMultipleParametersMouseDrag, 1272
 - HandleMultipleParametersMouseUp, 1273
 - HandleParameterMouseDown, 1270
 - HandleParameterMouseDrag, 1270
 - HandleParameterMouseEnter, 1271
 - HandleParameterMouseExit, 1271
 - HandleParameterMouseUp, 1271
 - SetViewSize, 1269
- AAX_VViewContainer.h, 1698
- AAX_WORD_ALIGNED_HINT
 - AAX_MiscUtils.h, 1716
- AAXATTR_Client_Level
 - AAX_UIDs.h, 1668
- AAXATTR_ClientFeature_AuxOutputStem
 - AAX_UIDs.h, 1667
- AAXATTR_ClientFeature_MIDI
 - AAX_UIDs.h, 1667
- AAXATTR_ClientFeature_SideChainInput
 - AAX_UIDs.h, 1667
- AAXATTR_ClientFeature_StemFormat
 - AAX_UIDs.h, 1667
- AAXCanUnloadNow
 - AAX_Init.h, 1576
- AAXCompID_AAXCollection
 - AAX_UIDs.h, 1655
- AAXCompID_AAXComponentDescriptor

- AAX_UIDs.h, [1656](#)
- AAXComplID_AAXEffectDescriptor
 - AAX_UIDs.h, [1656](#)
- AAXComplID_AAXPropertyMap
 - AAX_UIDs.h, [1657](#)
- AAXComplID_AutomationDelegate
 - AAX_UIDs.h, [1658](#)
- AAXComplID_Controller
 - AAX_UIDs.h, [1658](#)
- AAXComplID_DataBuffer
 - AAX_UIDs.h, [1666](#)
- AAXComplID_EffectDirectData
 - AAX_UIDs.h, [1665](#)
- AAXComplID_EffectGUI
 - AAX_UIDs.h, [1665](#)
- AAXComplID_EffectParameters
 - AAX_UIDs.h, [1664](#)
- AAXComplID_HostProcessor
 - AAX_UIDs.h, [1664](#)
- AAXComplID_HostProcessorDelegate
 - AAX_UIDs.h, [1657](#)
- AAXComplID_HostServices
 - AAX_UIDs.h, [1655](#)
- AAXComplID_PageTable
 - AAX_UIDs.h, [1662](#)
- AAXComplID_PageTableController
 - AAX_UIDs.h, [1659](#)
- AAXComplID_PrivateDataAccess
 - AAX_UIDs.h, [1660](#)
- AAXComplID_SessionDocument
 - AAX_UIDs.h, [1663](#)
- AAXComplID_SessionDocumentClient
 - AAX_UIDs.h, [1666](#)
- AAXComplID_Task
 - AAX_UIDs.h, [1663](#)
- AAXComplID_TaskAgent
 - AAX_UIDs.h, [1666](#)
- AAXComplID_Transport
 - AAX_UIDs.h, [1661](#)
- AAXComplID_TransportControl
 - AAX_UIDs.h, [1661](#)
- AAXComplID_ViewContainer
 - AAX_UIDs.h, [1660](#)
- AAXCreateObjectProc
 - AAX_Callbacks.h, [1339](#)
- AAXGetClassFactory
 - AAX_Init.h, [1576](#)
- AAXGetSDKVersion
 - AAX_Init.h, [1578](#)
- AAXLibrary_AAX_StringUtilities_h
 - AAX_StringUtilities.h, [1636](#)
- AAXPointer_32bit
 - AAX.h, [1309](#)
- AAXPointer_64bit
 - AAX.h, [1309](#)
- AAXRegisterComponent
 - AAX_Init.h, [1575](#)
- AAXRegisterPlugin
 - Description callback, [65](#)
- AAXShutdown
 - AAX_Init.h, [1577](#)
- AAXStartup
 - AAX_Init.h, [1577](#)
- AbsMax
 - AAX, [395](#)
- Accepts
 - AAX_IMIDIMessageInfoDelegate, [1045](#)
- Accepts_ExactStatus
 - AAX_IMIDIMessageInfoDelegate, [1046](#)
- ACF Elements, [149](#)
- ACF_DECLARE_STANDARD_UNKNOWN
 - AAX_IDataBuffer, [987](#)
 - AAX_IEffectDirectData, [1023](#)
 - AAX_IEffectGUI, [1026](#)
 - AAX_IEffectParameters, [1033](#)
 - AAX_IHostProcessor, [1037](#)
 - AAX_ISessionDocumentClient, [1104](#)
 - AAX_ITaskAgent, [1118](#)
- ACFCanUnloadNow
 - AAX_Exports.cpp, [1512](#)
- ACFGetClassFactory
 - AAX_Exports.cpp, [1511](#)
- ACFGetSDKVersion
 - AAX_Exports.cpp, [1513](#)
- acfIID
 - AAX_ACFInterface.doxygen, [1293](#)
- ACFRegisterComponent
 - AAX_Exports.cpp, [1511](#)
- ACFRegisterPlugin
 - AAX_Exports.cpp, [1511](#)
- ACFShutdown
 - AAX_Exports.cpp, [1513](#)
- ACFStartup
 - AAX_Exports.cpp, [1512](#)
- acfUID
 - AAX.h, [1315](#)
 - AAX_ACFInterface.doxygen, [1293](#)
- Acquire
 - AAX_VPropertyMap, [1246](#)
- AcquireFeatureProperties
 - AAX_IACFDescriptionHost, [786](#)
 - AAX_IDescriptionHost, [990](#)
 - AAX_VDescriptionHost, [1211](#)
- AcquireProperties
 - AAX_IACFFeatureInfo, [849](#)
 - AAX_IFeatureInfo, [1034](#)
 - AAX_VFeatureInfo, [1221](#)
- Add
 - AAX_CStringAbbreviations, [708](#)
- AddAcceptedResult
 - AAX_CheckedResult, [509](#)
- AddAudioBufferLength
 - AAX_IACFComponentDescriptor, [753](#)
 - AAX_IComponentDescriptor, [955](#)
 - AAX_VComponentDescriptor, [1180](#)
- AddAudioIn

- AAX_IACFComponentDescriptor, [752](#)
- AAX_IComponentDescriptor, [953](#)
- AAX_VComponentDescriptor, [1179](#)
- AddAudioOut
 - AAX_IACFComponentDescriptor, [752](#)
 - AAX_IComponentDescriptor, [954](#)
 - AAX_VComponentDescriptor, [1179](#)
- AddAuxOutputStem
 - AAX_IACFComponentDescriptor, [755](#)
 - AAX_IComponentDescriptor, [958](#)
 - AAX_VComponentDescriptor, [1182](#)
- AddCategory
 - AAX_IACFEffectorDescriptor, [789](#)
 - AAX_IEffectDescriptor, [1017](#)
 - AAX_VEffectDescriptor, [1216](#)
- AddCategoryBypassParameter
 - AAX_IACFEffectorDescriptor, [789](#)
 - AAX_IEffectDescriptor, [1017](#)
 - AAX_VEffectDescriptor, [1216](#)
- AddClock
 - AAX_IACFComponentDescriptor, [754](#)
 - AAX_IComponentDescriptor, [956](#)
 - AAX_VComponentDescriptor, [1180](#)
- AddComponent
 - AAX_IACFEffectorDescriptor, [788](#)
 - AAX_IEffectDescriptor, [1016](#)
 - AAX_VEffectDescriptor, [1215](#)
- AddControlMIDIINode
 - AAX_IACFEffectorDescriptor_V2, [793](#)
 - AAX_IEffectDescriptor, [1019](#)
 - AAX_VEffectDescriptor, [1218](#)
- AddDataInPort
 - AAX_IACFComponentDescriptor, [754](#)
 - AAX_IComponentDescriptor, [957](#)
 - AAX_VComponentDescriptor, [1181](#)
- AddDmaInstance
 - AAX_IACFComponentDescriptor, [756](#)
 - AAX_IComponentDescriptor, [960](#)
 - AAX_VComponentDescriptor, [1184](#)
- AddDouble
 - AAX_CChunkDataParser, [442](#)
- AddEffect
 - AAX_IACFCollection, [748](#)
 - AAX_ICollection, [948](#)
 - AAX_VCollection, [1170](#)
- AddFloat
 - AAX_CChunkDataParser, [442](#)
- AddInt16
 - AAX_CChunkDataParser, [442](#)
- AddInt32
 - AAX_CChunkDataParser, [442](#)
- Additional AAX features, [80](#)
- Additional Topics, [111](#)
- AddMeterDescription
 - AAX_IACFEffectorDescriptor, [791](#)
 - AAX_IEffectDescriptor, [1019](#)
 - AAX_VEffectDescriptor, [1218](#)
- AddMeters
 - AAX_IACFComponentDescriptor, [759](#)
 - AAX_IComponentDescriptor, [961](#)
 - AAX_VComponentDescriptor, [1184](#)
- AddMIDIINode
 - AAX_IACFComponentDescriptor, [757](#)
 - AAX_IComponentDescriptor, [962](#)
 - AAX_VComponentDescriptor, [1186](#)
- AddName
 - AAX_IACFEffectorDescriptor, [788](#)
 - AAX_IEffectDescriptor, [1017](#)
 - AAX_VEffectDescriptor, [1215](#)
- AddPackageName
 - AAX_IACFCollection, [748](#)
 - AAX_ICollection, [948](#)
 - AAX_VCollection, [1171](#)
- AddParameter
 - AAX_CParameterManager, [620](#)
- AddPointerProperty
 - AAX_IPropertyMap, [1097](#), [1098](#)
 - AAX_VPropertyMap, [1247](#), [1248](#)
- AddPrivateData
 - AAX_IACFComponentDescriptor, [756](#)
 - AAX_IComponentDescriptor, [959](#)
 - AAX_VComponentDescriptor, [1183](#)
- AddProcessProc
 - AAX_IACFComponentDescriptor_V3, [765](#)
 - AAX_IComponentDescriptor, [966](#)
 - AAX_VComponentDescriptor, [1188](#)
- AddProcessProc_Native
 - AAX_IACFComponentDescriptor, [757](#)
 - AAX_IComponentDescriptor, [964](#), [967](#)
 - AAX_VComponentDescriptor, [1187](#)
- AddProcessProc_Tl
 - AAX_IACFComponentDescriptor, [758](#)
 - AAX_IComponentDescriptor, [965](#)
 - AAX_VComponentDescriptor, [1188](#)
- AddProcPtr
 - AAX_IACFEffectorDescriptor, [789](#)
 - AAX_IEffectDescriptor, [1018](#)
 - AAX_VEffectDescriptor, [1217](#)
- AddProperty
 - AAX_IACFPropertyMap, [895](#)
 - AAX_IPropertyMap, [1097](#)
 - AAX_VPropertyMap, [1247](#)
- AddProperty64
 - AAX_IACFPropertyMap_V3, [900](#)
- AddPropertyWithIDArray
 - AAX_IACFPropertyMap_V2, [897](#)
 - AAX_IPropertyMap, [1099](#)
 - AAX_VPropertyMap, [1248](#)
- AddRef
 - IACFUnknown, [1283](#)
- AddReservedField
 - AAX_IACFComponentDescriptor, [752](#)
 - AAX_IComponentDescriptor, [963](#)
 - AAX_VComponentDescriptor, [1178](#)
- AddResourceInfo
 - AAX_IACFEffectorDescriptor, [791](#)

- AAX_IEffectDescriptor, 1019
- AAX_VEffectDescriptor, 1217
- AddResult
 - AAX_IACFTask, 908
 - AAX_ITask, 1115
 - AAX_VTask, 1255
- AddSampleRate
 - AAX_IACFComponentDescriptor, 753
 - AAX_IComponentDescriptor, 955
 - AAX_VComponentDescriptor, 1180
- AddShortenedName
 - AAX_CParameter< T >, 586
 - AAX_CStatelessParameter, 664
 - AAX_IParameter, 1068
- AddShortenedStrings
 - AAX_CBinaryDisplayDelegate< T >, 434
 - AAX_CStateDisplayDelegate< T >, 658
- AddSideChainIn
 - AAX_IACFComponentDescriptor, 754
 - AAX_IComponentDescriptor, 957
 - AAX_VComponentDescriptor, 1181
- AddString
 - AAX_CChunkDataParser, 442
- AddSynchronizedParameter
 - AAX_CMonolithicParameters, 551
- AddTask
 - AAX_CTaskAgent, 726, 727
 - AAX_IACFTaskAgent, 911
- AddTemporaryData
 - AAX_IACFComponentDescriptor_V2, 762
 - AAX_IComponentDescriptor, 960
 - AAX_VComponentDescriptor, 1183
- alignFree
 - AAX, 390
- alignMalloc
 - AAX, 391
- AnalyzeAudio
 - AAX_CHostProcessor, 519
 - AAX_IACFHostProcessor, 855
- Any
 - AAX::Exception::Any, 1275
- Append
 - AAX_CString, 697
- AppendHex
 - AAX_CString, 698
- AppendNumber
 - AAX_CString, 698
- Assert
 - AAX_IACFHostServices, 867
- AsString
 - AAX, 377
- AsStringFourChar
 - AAX, 385
- AsStringIDTriad
 - AAX, 387
- AsStringInt32
 - AAX, 387
- AsStringMIDIStream_Debug
 - Other Extensions, 284
- AsStringPropertyValue
 - AAX, 386
- AsStringResult
 - AAX, 388
- AsStringStemChannel
 - AAX, 388
- AsStringStemFormat
 - AAX, 387
- AsStringUInt32
 - AAX, 387
- AsUnknown
 - AAX_VPageTable, 1240
- Automatable
 - AAX_CParameter< T >, 600
 - AAX_CStatelessParameter, 666
 - AAX_IParameter, 1068
- AutomationDelegate
 - AAX_CEffectParameters, 503
- Auxiliary Output Stems, 93
- Background processing callback, 96
- Basic parameter update sequences, 127
- Binary2String
 - AAX, 383
- BoolToNormalized
 - AAX_CEffectParameters.h, 1364
- BUILD_DATA_FAILED
 - AAX_ChunkDataParserDefs, 410
- BuildChunkData
 - AAX_CEffectParameters, 505
- Call
 - AAX_CPacketHandler< TWorker >, 574
 - AAX_IPacketHandler, 1051
- CancelAllTasks
 - AAX_CTaskAgent, 726
 - AAX_IACFTaskAgent, 911
- Canceled
 - Task agent interface, 104
- Caseless_strcmp
 - AAX, 383
- CBackgroundProc
 - AAX_Component< aContextType >, 566
- cBigEndian
 - AAX, 402
- cDefaultMasterBypassID
 - AAX_CEffectParameters.h, 1365
- cDenormalAvoidanceOffset
 - AAX, 405
- CeilLog2
 - AAX, 396
- cFloatDenormalAvoidanceOffset
 - AAX, 405
- cGiga
 - AAX, 405
- cHalfPi
 - AAX, 403
- Change Log, 334

- Check
 - AAX_AggregateResult, [414](#)
- cInitialSeedValue
 - AAX, [406](#)
- CInitPrivateDataProc
 - AAX_Component< aContextType >, [566](#)
- CInstanceInitProc
 - AAX_Component< aContextType >, [566](#)
- cKilo
 - AAX, [405](#)
- ClampToZero
 - AAX, [392](#)
- Clear
 - AAX_AggregateResult, [414](#)
 - AAX_CAtomicQueue< T, S >, [427](#)
 - AAX_CChunkDataParser, [444](#)
 - AAX_CheckedResult, [511](#)
 - AAX_CString, [696](#)
 - AAX_CStringAbbreviations, [709](#)
 - AAX_IACFComponentDescriptor, [751](#)
 - AAX_IACFPageTable, [874](#)
 - AAX_IComponentDescriptor, [953](#)
 - AAX_IContainer, [969](#)
 - AAX_IPageTable, [1053](#)
 - AAX_IPointerQueue< T >, [1091](#)
 - AAX_VComponentDescriptor, [1178](#)
 - AAX_VPageTable, [1231](#)
 - AAX_VSessionDocument, [1251](#)
- ClearMappedParameter
 - AAX_IACFPageTable, [876](#)
 - AAX_IPageTable, [1056](#)
 - AAX_VPageTable, [1233](#)
- ClearMappedParameterByID
 - AAX, [382](#)
- ClearMeterClipped
 - AAX_IACFController, [774](#)
 - AAX_IController, [981](#)
 - AAX_VController, [1202](#)
- ClearMeterPeakValue
 - AAX_IACFController, [773](#)
 - AAX_IController, [979](#)
 - AAX_VController, [1201](#)
- ClearNameVariationsForParameter
 - AAX_IACFPageTable_V2, [883](#)
 - AAX_IPageTable, [1061](#)
 - AAX_VPageTable, [1238](#)
- ClearParameterNameVariations
 - AAX_IACFPageTable_V2, [883](#)
 - AAX_IPageTable, [1061](#)
 - AAX_VPageTable, [1238](#)
- ClearShortenedNames
 - AAX_CParameter< T >, [587](#)
 - AAX_CStatelessParameter, [665](#)
 - AAX_IParameter, [1068](#)
- cLittleEndian
 - AAX, [402](#)
- Clone
 - AAX_CBinaryDisplayDelegate< T >, [432](#)
 - AAX_CBinaryTaperDelegate< T >, [437](#)
 - AAX_CDecibelDisplayDelegateDecorator< T >, [449](#)
 - AAX_CLinearTaperDelegate< T, RealPrecision >, [530](#)
 - AAX_CLogTaperDelegate< T, RealPrecision >, [535](#)
 - AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >, [562](#)
 - AAX_CPacketHandler< TWorker >, [574](#)
 - AAX_CParameterValue< T >, [625](#)
 - AAX_CPercentDisplayDelegateDecorator< T >, [633](#)
 - AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, [639](#)
 - AAX_CRangeTaperDelegate< T, RealPrecision >, [644](#)
 - AAX_CStateDisplayDelegate< T >, [656](#)
 - AAX_CStateTaperDelegate< T >, [688](#)
 - AAX_CStringDisplayDelegate< T >, [720](#)
 - AAX_CUnitDisplayDelegateDecorator< T >, [731](#)
 - AAX_CUnitPrefixDisplayDelegateDecorator< T >, [737](#)
 - AAX_IDisplayDelegate< T >, [994](#)
 - AAX_IDisplayDelegateDecorator< T >, [1000](#)
 - AAX_IPacketHandler, [1051](#)
 - AAX_IParameterValue, [1086](#)
 - AAX_ITaperDelegate< T >, [1108](#)
- CloneValue
 - AAX_CParameter< T >, [585](#)
 - AAX_CStatelessParameter, [662](#)
 - AAX_IParameter, [1066](#)
- cMega
 - AAX, [405](#)
- cMicro
 - AAX, [404](#)
- cMilli
 - AAX, [404](#)
- cNano
 - AAX, [405](#)
- cNeg3dB
 - AAX, [404](#)
- cNeg6dB
 - AAX, [404](#)
- cNormalizeLongToAmplitudeOne
 - AAX, [404](#)
- cNormalizeLongToAmplitudeOneHalf
 - AAX, [404](#)
- CoarseSampleRate
 - AAX_SampleRateUtils.h, [1726](#)
- CoarseSampleRateFactor
 - AAX_SampleRateUtils.h, [1727](#)
- CoarseSampleRateIndex
 - AAX_SampleRateUtils.h, [1727](#)
- Compare
 - AAX_CStateDisplayDelegate< T >, [658](#)
- CompareActiveChunk
 - AAX_CEffectParameters, [495](#)

- AAX_IACEffectParameters, [828](#)
- cOneOverRootTwo
 - AAX, [403](#)
- ConstrainRealValue
 - AAX_CBinaryTaperDelegate< T >, [437](#)
 - AAX_CLinearTaperDelegate< T, RealPrecision >, [531](#)
 - AAX_CLogTaperDelegate< T, RealPrecision >, [536](#)
 - AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, [639](#)
 - AAX_CRangeTaperDelegate< T, RealPrecision >, [645](#)
 - AAX_CStateTaperDelegate< T >, [688](#)
 - AAX_ITaperDelegate< T >, [1109](#)
- Controller
 - AAX_CEffectDirectData, [457](#)
 - AAX_CEffectParameters, [502](#)
 - AAX_CHostProcessor, [524](#)
- CopyAttribute
 - IACFDefinition, [1281](#)
- CopyPageTable
 - AAX, [381](#)
- CopyTableForEffect
 - AAX_IACFPageTableController, [886](#)
- CopyTableForEffectFromFile
 - AAX_IACFPageTableController_V2, [889](#)
- CopyTableOfLayoutForEffect
 - AAX_IACFPageTableController, [887](#)
- CopyTableOfLayoutFromFile
 - AAX_IACFPageTableController_V2, [890](#)
- Core AAX Interface, [55](#)
- CPacketAllocator
 - AAX_Component< aContextType >, [566](#)
- cPi
 - AAX, [403](#)
- cPico
 - AAX, [405](#)
- cPos3dB
 - AAX, [403](#)
- cPos6dB
 - AAX, [404](#)
- cPreviewID
 - AAX_CEffectParameters.h, [1365](#)
- CProcessProc
 - AAX_Component< aContextType >, [565](#)
- cQuarterPi
 - AAX, [403](#)
- Create
 - AAX_VPropertyMap, [1245](#)
- CreateTableCopyForEffect
 - AAX_IController, [983](#)
 - AAX_VController, [1204](#)
- CreateTableCopyForEffectFromFile
 - AAX_IController, [984](#)
 - AAX_VController, [1205](#)
- CreateTableCopyForLayout
 - AAX_IController, [983](#)
- AAX_VController, [1205](#)
- CreateTableCopyForLayoutFromFile
 - AAX_IController, [985](#)
- AAX_VController, [1206](#)
- CreateViewContainer
 - AAX_CEffectGUI, [466](#)
- CreateViewContents
 - AAX_CEffectGUI, [466](#)
- cRootTwo
 - AAX, [403](#)
- cSeedDivisor
 - AAX, [406](#)
- CString
 - AAX_CString, [700](#)
- cTwoPi
 - AAX, [403](#)
- Data
 - AAX_CArrayDataBuffer< D >, [420](#)
 - AAX_CArrayDataBufferOfType< T, D >, [424](#)
 - AAX_CStringDataBuffer, [713](#)
 - AAX_CStringDataBufferOfType< T >, [717](#)
 - AAX_IACFDataBuffer, [784](#)
 - AAX_IDataBufferWrapper, [989](#)
 - AAX_ISessionDocument::TempoMap, [1289](#)
 - AAX_VDataBufferWrapper, [1209](#)
 - AAX_VSessionDocument::VTempoMap, [1291](#)
- Data model interface, [72](#)
- Data1
 - _acfUID, [411](#)
- Data2
 - _acfUID, [411](#)
- Data3
 - _acfUID, [411](#)
- Data4
 - _acfUID, [411](#)
- DataValue
 - AAX_CChunkDataParser::DataValue, [1277](#)
- DBToGain
 - AAX_CommonConversions.h, [1377](#)
- DeDenormal
 - AAX, [391](#)
- DeDenormalFine
 - AAX, [391](#)
- DEFAULT32BIT_TYPE_INCR
 - AAX_ChunkDataParserDefs, [410](#)
- DEFAULT32BIT_TYPE_SIZE
 - AAX_ChunkDataParserDefs, [409](#)
- Defaults
 - AAX_CParameter< T >, [580](#)
 - AAX_CParameterValue< T >, [623](#)
- DEFINE_AAX_ERROR_STRING
 - AAX_StringUtilities.hpp, [1638](#)
- DefineAttribute
 - IACFDefinition, [1280](#)
- DeleteViewContainer
 - AAX_CEffectGUI, [466](#)
- Desc
 - AAX::Exception::Any, [1276](#)

- Description callback, [56](#)
 - AAXRegisterPlugin, [65](#)
 - GetEffectDescriptions, [65](#)
- DescriptionHost
 - AAX_ICollection, [950](#)
 - AAX_VCollection, [1173](#), [1174](#)
 - AAX_VDescriptionHost, [1212](#)
- DigiTrace Guide, [260](#)
- Direct data access interface, [81](#)
- Direct Memory Access, [94](#)
- Dispatch
 - AAX_CPacketDispatcher, [571](#)
- Display delegate decorators, [110](#)
- Display delegates, [109](#)
- DisplayDelegate
 - AAX_CParameter< T >, [607](#)
- Distributing Your AAX Plug-In, [290](#)
- DoMIDITransfers
 - AAX_CEffectParameters, [500](#)
 - AAX_IACFEffEffectParameters, [830](#)
- Done
 - Task agent interface, [104](#)
- DoTableLookupExtraFast
 - AAX_FastInterpolatedTableLookup< DFLOAT >, [741](#), [742](#)
- DoTableLookupExtraFastMulti
 - AAX_FastInterpolatedTableLookup< DFLOAT >, [742](#)
- DOUBLE_STRING_IDENTIFIER
 - AAX_ChunkDataParserDefs, [408](#)
- DOUBLE_TYPE
 - AAX_ChunkDataParserDefs, [408](#)
- DOUBLE_TYPE_INCR
 - AAX_ChunkDataParserDefs, [408](#)
- DOUBLE_TYPE_SIZE
 - AAX_ChunkDataParserDefs, [408](#)
- DoubleTo32BitDSPCoef
 - AAX_CommonConversions.h, [1380](#)
- DoubleTo32BitDSPCoefRnd
 - AAX_CommonConversions.h, [1379](#)
- DoubleToDSPCoef
 - AAX_CommonConversions.h, [1378](#)
- DoubleToDSPCoefRnd
 - AAX_CommonConversions.h, [1380](#)
- DoubleToLong
 - AAX_CommonConversions.h, [1378](#)
- DoubleToLongControl
 - AAX_SliderConversions.h, [1633](#)
- DoubleToLongControlNonlinear
 - AAX_SliderConversions.h, [1633](#)
- Draw
 - AAX_CEffectGUI, [464](#)
 - AAX_IACFEffEffectGUI, [804](#)
- DSH Guide, [271](#)
- DSH_Guide.doxygen, [1296](#)
- DSPCoefToDouble
 - AAX_CommonConversions.h, [1378](#)
- DTT Guide, [277](#)
- DTT_Guide.doxygen, [1296](#)
- DuplicatePropertyMap
 - AAX_IComponentDescriptor, [964](#)
 - AAX_VComponentDescriptor, [1187](#)
- e176400SampleRate
 - AAX, [377](#)
- e192000SampleRate
 - AAX, [377](#)
- e44100SampleRate
 - AAX, [377](#)
- e48000SampleRate
 - AAX, [377](#)
- e88200SampleRate
 - AAX, [377](#)
- e96000SampleRate
 - AAX, [377](#)
- EChannelModeData
 - AAX, [376](#)
- eChannelModeData_AllNotesOff
 - AAX, [376](#)
- eChannelModeData_AllSoundOff
 - AAX, [376](#)
- eChannelModeData_LocalControl
 - AAX, [376](#)
- eChannelModeData_OmniOff
 - AAX, [376](#)
- eChannelModeData_OmniOn
 - AAX, [376](#)
- eChannelModeData_PolyOff
 - AAX, [376](#)
- eChannelModeData_PolyOn
 - AAX, [376](#)
- eChannelModeData_ResetControllers
 - AAX, [376](#)
- EffectInit
 - AAX_CEffectParameters, [504](#)
- EffectParameters
 - AAX_CEffectDirectData, [457](#)
 - AAX_CHostProcessor, [525](#)
- EMode
 - AAX_IDma, [1005](#)
- eMode_Burst
 - AAX_IDma, [1007](#)
- eMode_Error
 - AAX_IDma, [1007](#)
- eMode_Gather
 - AAX_IDma, [1007](#)
- eMode_Scatter
 - AAX_IDma, [1007](#)
- Empty
 - AAX_CString, [696](#)
 - AAX_IACFPPageTable, [874](#)
 - AAX_IPageTable, [1053](#)
 - AAX_VPageTable, [1231](#)
- ENDIANSWAP_H
 - AAX_EndianSwap.h, [1430](#)
- eParameterDefaultMaxIdentifierLength
 - AAX_CParameterValue< T >, [623](#)

- eParameterDefaultMaxIdentifierSize
 - AAX_CParameterValue< T >, 623
- eParameterDefaultNumStepsContinuous
 - AAX_CParameter< T >, 581
- eParameterDefaultNumStepsDiscrete
 - AAX_CParameter< T >, 581
- eParameterTypeBool
 - AAX_CParameter< T >, 580
- eParameterTypeCustom
 - AAX_CParameter< T >, 580
- eParameterTypeFloat
 - AAX_CParameter< T >, 580
- eParameterTypeInt32
 - AAX_CParameter< T >, 580
- eParameterTypeUndefined
 - AAX_CParameter< T >, 580
- EQ and Dynamics Curve Displays, 98
 - AAX_ECType, 100
 - AAX_eCurveType_Dynamics, 101
 - AAX_eCurveType_EQ, 101
 - AAX_eCurveType_None, 101
 - AAX_eCurveType_Reduction, 101
 - GetCurveData, 101
 - GetCurveDataDisplayRange, 102
 - GetCurveDataMeterIds, 102
- Equals
 - AAX_CString, 702, 703
- Erase
 - AAX_CString, 697
- Error
 - Task agent interface, 104
- ESampleRates
 - AAX, 377
- ESpecialData
 - AAX, 376
- eSpecialData_AccentedClick
 - AAX, 376
- eSpecialData_UnaccentedClick
 - AAX, 376
- ESRUtils
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_192kIndex
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_192kRangeCoarse
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_192kRangeMax
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_192kRangeMin
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_48kIndex
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_48kRangeCoarse
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_48kRangeMax
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_48kRangeMin
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_96kIndex
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_96kRangeCoarse
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_96kRangeMax
 - AAX_SampleRateUtils.h, 1726
- eSRUtils_96kRangeMin
 - AAX_SampleRateUtils.h, 1726
- EState
 - AAX_IDma, 1005
- eState_Complete
 - AAX_IDma, 1005
- eState_Error
 - AAX_IDma, 1005
- eState_Init
 - AAX_IDma, 1005
- eState_Pending
 - AAX_IDma, 1005
- eState_Running
 - AAX_IDma, 1005
- EStatus
 - AAX_IContainer, 969
- eStatus_NotInitialized
 - AAX_IContainer, 969
- eStatus_Overflow
 - AAX_IContainer, 969
- eStatus_Success
 - AAX_IContainer, 969
- eStatus_Unavailable
 - AAX_IContainer, 969
- eStatus_Unsupported
 - AAX_IContainer, 969
- EStatusByte
 - AAX, 375
- eStatusByte_ActiveSensing
 - AAX, 376
- eStatusByte_Continue
 - AAX, 376
- eStatusByte_MTCQuarterFrame
 - AAX, 376
- eStatusByte_Reset
 - AAX, 376
- eStatusByte_SongPosition
 - AAX, 376
- eStatusByte_SongSelect
 - AAX, 376
- eStatusByte_Start
 - AAX, 376
- eStatusByte_Stop
 - AAX, 376
- eStatusByte_SysExBegin
 - AAX, 375
- eStatusByte_SysExEnd
 - AAX, 376
- eStatusByte_TimingClock
 - AAX, 376
- eStatusByte_TuneRequest
 - AAX, 376
- EStatusNibble

- AAX, [375](#)
- eStatusNibble_ChannelMode
 - AAX, [375](#)
- eStatusNibble_ChannelPressure
 - AAX, [375](#)
- eStatusNibble_ControlChange
 - AAX, [375](#)
- eStatusNibble_KeyPressure
 - AAX, [375](#)
- eStatusNibble_NoteOff
 - AAX, [375](#)
- eStatusNibble_NoteOn
 - AAX, [375](#)
- eStatusNibble_PitchBend
 - AAX, [375](#)
- eStatusNibble_ProgramChange
 - AAX, [375](#)
- eStatusNibble_SystemCommon
 - AAX, [375](#)
- eStatusNibble_SystemRealTime
 - AAX, [375](#)
- Example Plug-Ins, [352](#)
- Exception
 - AAX_CheckedResult, [508](#)
- Extensions, [281](#)
- fabs
 - AAX, [394](#)
- fabsf
 - AAX, [394](#)
- FastRndDbl2Int32
 - AAX, [397](#)
- FastRound2Int32
 - AAX, [396](#), [397](#)
- FastRound2Int64
 - AAX, [399](#)
- FastTrunc2Int32
 - AAX, [398](#), [399](#)
- fChunkID
 - AAX_SPlugInChunk, [1155](#)
 - AAX_SPlugInChunkHeader, [1157](#)
- fData
 - AAX_SPlugInChunk, [1155](#)
- Fill
 - AAX, [392](#), [393](#)
- FilterDenormals
 - AAX, [391](#)
- FilterParameterIDOnSave
 - AAX_CEffectParameters, [504](#)
- FindDouble
 - AAX_CChunkDataParser, [443](#)
- FindFirst
 - AAX_CString, [700](#)
- FindFloat
 - AAX_CChunkDataParser, [443](#)
- FindInt16
 - AAX_CChunkDataParser, [443](#)
- FindInt32
 - AAX_CChunkDataParser, [443](#)
- FindLast
 - AAX_CString, [700](#)
- FindName
 - AAX_CChunkDataParser, [445](#)
- FindParameterMappingsInPageTable
 - AAX, [382](#)
- FindString
 - AAX_CChunkDataParser, [443](#)
- FLOAT_STRING_IDENTIFIER
 - AAX_ChunkDataParserDefs, [408](#)
- FLOAT_TYPE
 - AAX_ChunkDataParserDefs, [408](#)
- fManufacturerID
 - AAX_SPlugInChunk, [1154](#)
 - AAX_SPlugInChunkHeader, [1157](#)
- fName
 - AAX_SPlugInChunk, [1155](#)
 - AAX_SPlugInChunkHeader, [1158](#)
- ForceAnalyze
 - AAX_IACFHostProcessorDelegate_V2, [863](#)
 - AAX_IHostProcessorDelegate, [1040](#)
 - AAX_VHostProcessorDelegate, [1224](#)
- ForceProcess
 - AAX_IACFHostProcessorDelegate_V3, [865](#)
 - AAX_IHostProcessorDelegate, [1040](#)
 - AAX_VHostProcessorDelegate, [1224](#)
- FormatResult
 - AAX::Exception::ResultError, [1286](#)
- fPlugInID
 - AAX_SPlugInChunk, [1155](#)
 - AAX_SPlugInChunkHeader, [1157](#)
- fProductID
 - AAX_SPlugInChunk, [1154](#)
 - AAX_SPlugInChunkHeader, [1157](#)
- fpt
 - AAX_CPacketHandler< TWorker >, [574](#)
- fptEx
 - AAX_CPacketHandler< TWorker >, [574](#)
- fSize
 - AAX_SPlugInChunk, [1154](#)
 - AAX_SPlugInChunkHeader, [1157](#)
- Function
 - AAX::Exception::Any, [1276](#)
- fVersion
 - AAX_SPlugInChunk, [1154](#)
 - AAX_SPlugInChunkHeader, [1157](#)
- GainToDB
 - AAX_CommonConversions.h, [1377](#)
- GenerateCoefficients
 - AAX_CEffectParameters, [491](#)
 - AAX_CMonolithicParameters, [553](#)
 - AAX_IACFEffEffectParameters, [825](#)
- GenerateSingleValuePacket
 - AAX_CPacketDispatcher, [571](#)
- Get
 - AAX_CParameterValue< T >, [625](#)
 - AAX_CString, [694](#)
 - AAX_CStringAbbreviations, [708](#)

- AAX_IString, [1106](#)
 - SAutoArray< T >, [1288](#)
- GetArgumentOfType
 - AAX_IACFTask, [907](#)
 - AAX_ITask, [1114](#)
 - AAX_VTask, [1254](#)
- GetAttributeInfo
 - IACFDefinition, [1281](#)
- GetAudio
 - AAX_CHostProcessor, [524](#)
 - AAX_IACFHostProcessorDelegate, [860](#)
 - AAX_IHostProcessorDelegate, [1039](#)
 - AAX_VHostProcessorDelegate, [1223](#)
- GetBarBeatPosition
 - AAX_IACFTransport, [916](#)
 - AAX_ITransport, [1124](#)
 - AAX_VTransport, [1262](#)
- GetBaseOffset
 - AAX_IDma, [1013](#)
- GetBoolFromNormalizedValue
 - AAX_CParameter< T >, [596](#), [611](#)
 - AAX_CStatelessParameter, [674](#)
 - AAX_IParameter, [1075](#)
- GetBurstLength
 - AAX_IDma, [1010](#)
- GetChunk
 - AAX_CEffectParameters, [494](#)
 - AAX_IACFEffEffectParameters, [827](#)
- GetChunkData
 - AAX_CChunkDataParser, [444](#)
- GetChunkDataSize
 - AAX_CChunkDataParser, [444](#)
- GetChunkIDFromIndex
 - AAX_CEffectParameters, [493](#)
 - AAX_IACFEffEffectParameters, [826](#)
- GetChunkSize
 - AAX_CEffectParameters, [493](#)
 - AAX_IACFEffEffectParameters, [826](#)
- GetChunkVersion
 - AAX_CChunkDataParser, [444](#)
- GetClipNameSuffix
 - AAX_CHostProcessor, [521](#)
 - AAX_IACFHostProcessor_V2, [859](#)
- GetCoefficient
 - AAX_Map, [1135](#)
- GetController
 - AAX_CEffectGUI, [467](#)
 - AAX_CSessionDocumentClient, [652](#)
 - AAX_CTaskAgent, [727](#)
- GetCStringOfLength
 - AAX, [383](#)
- GetCurrentAutomationTimestamp
 - AAX_IACFController_V2, [778](#)
 - AAX_IController, [981](#)
 - AAX_VController, [1196](#)
- GetCurrentLoopPosition
 - AAX_IACFTransport, [915](#)
 - AAX_ITransport, [1122](#)
- AAX_VTransport, [1261](#)
- GetCurrentMeter
 - AAX_IACFTransport, [914](#)
 - AAX_ITransport, [1121](#)
 - AAX_VTransport, [1260](#)
- GetCurrentMeterValue
 - AAX_IACFController, [772](#)
 - AAX_IController, [978](#)
 - AAX_VController, [1201](#)
- GetCurrentNativeSampleLocation
 - AAX_IACFTransport, [915](#)
 - AAX_ITransport, [1123](#)
 - AAX_VTransport, [1261](#)
- GetCurrentTempo
 - AAX_IACFTransport, [913](#)
 - AAX_ITransport, [1120](#)
 - AAX_VTransport, [1259](#)
- GetCurrentTickPosition
 - AAX_IACFTransport, [914](#)
 - AAX_ITransport, [1122](#)
 - AAX_VTransport, [1260](#)
- GetCurrentTicksPerBeat
 - AAX_IACFTransport, [917](#)
 - AAX_ITransport, [1124](#)
 - AAX_VTransport, [1263](#)
- GetCurveData
 - AAX_CEffectParameters, [496](#)
 - EQ and Dynamics Curve Displays, [101](#)
- GetCurveDataDisplayRange
 - AAX_CEffectParameters, [498](#)
 - EQ and Dynamics Curve Displays, [102](#)
- GetCurveDataMeterIds
 - AAX_CEffectParameters, [497](#)
 - EQ and Dynamics Curve Displays, [102](#)
- GetCustomData
 - AAX_CEffectParameters, [499](#)
 - AAX_IACFEffEffectParameters, [829](#)
- GetCustomLabel
 - AAX_CEffectGUI, [465](#)
 - AAX_IACFEffEffectGUI, [805](#)
- GetCustomTickPosition
 - AAX_IACFTransport, [916](#)
 - AAX_ITransport, [1123](#)
 - AAX_VTransport, [1262](#)
- GetCycleCount
 - AAX_IACFController, [769](#)
 - AAX_IController, [974](#)
 - AAX_VController, [1195](#)
- GetDefaultValue
 - AAX_CParameter< T >, [606](#)
- GetDmaMode
 - AAX_IDma, [1008](#)
- GetDmaState
 - AAX_IDma, [1008](#)
- GetDocumentData
 - AAX_IACFSessionDocument, [902](#)
 - AAX_ISessionDocument, [1101](#)
 - AAX_VSessionDocument, [1252](#)

- GetDoubleFromNormalizedValue
 - AAX_CParameter< T >, [597](#), [612](#)
 - AAX_CStatelessParameter, [676](#)
 - AAX_IParameter, [1077](#)
- GetDst
 - AAX_IDma, [1009](#)
- GetDstEnd
 - AAX_CHostProcessor, [523](#)
- GetDstStart
 - AAX_CHostProcessor, [523](#)
- GetEffectDescriptions
 - Description callback, [65](#)
- GetEffectID
 - AAX_IACFController, [768](#)
 - AAX_IController, [972](#)
 - AAX_VController, [1193](#)
- GetEffectParameters
 - AAX_CEffectGUI, [467](#)
 - AAX_CHostProcessor, [521](#)
 - AAX_CSessionDocumentClient, [652](#)
 - AAX_CTaskAgent, [727](#)
- GetFastInt32RPDF
 - AAX, [400](#)
- GetFastRPDFWithAmplitudeOne
 - AAX, [401](#)
- GetFeetFramesInfo
 - AAX_IACFTransport_V2, [920](#)
 - AAX_ITransport, [1125](#)
 - AAX_VTransport, [1264](#)
- GetFifoBuffer
 - AAX_IDma, [1011](#)
- GetFifoSize
 - AAX_IDma, [1014](#)
- GetFirstX
 - AAX_Map, [1136](#)
- GetFirstY
 - AAX_Map, [1136](#)
- GetFloatFromNormalizedValue
 - AAX_CParameter< T >, [597](#), [612](#)
 - AAX_CStatelessParameter, [675](#)
 - AAX_IParameter, [1076](#)
- GetHDTIMECodeInfo
 - AAX_IACFTransport_V3, [924](#)
 - AAX_ITransport, [1126](#)
 - AAX_VTransport, [1265](#)
- GetHostName
 - AAX_IACFController_V2, [779](#)
 - AAX_IController, [982](#)
 - AAX_VController, [1197](#)
- GetHostProcessorDelegate
 - AAX_CHostProcessor, [521](#), [522](#)
- GetHybridSignalLatency
 - AAX_IACFController_V2, [778](#)
 - AAX_VController, [1194](#)
 - Hybrid Processing architecture, [86](#)
- GetID
 - AAX_CPacket, [568](#)
- GetInputRange
 - AAX_CHostProcessor, [522](#)
- GetInputStemFormat
 - AAX_IACFController, [768](#)
 - AAX_IController, [973](#)
 - AAX_VController, [1193](#)
- GetInt32FromNormalizedValue
 - AAX_CParameter< T >, [596](#), [612](#)
 - AAX_CStatelessParameter, [675](#)
 - AAX_IParameter, [1076](#)
- GetInt32RPDF
 - AAX, [400](#)
- GetIsAudioSuite
 - AAX_IACFController_V3, [782](#)
 - AAX_IController, [982](#)
 - AAX_VController, [1195](#)
- GetIsUnknown
 - AAX_IPropertyMap, [1099](#)
 - AAX_VCollection, [1174](#)
 - AAX_VComponentDescriptor, [1190](#)
 - AAX_VEffectDescriptor, [1219](#)
 - AAX_VPropertyMap, [1249](#)
- GetLastX
 - AAX_Map, [1136](#)
- GetLastY
 - AAX_Map, [1136](#)
- GetLinearBuffer
 - AAX_IDma, [1012](#)
- GetLocation
 - AAX_CHostProcessor, [522](#)
- getLowestSampleRateInMask
 - AAX.h, [1317](#)
- GetMappedParameterID
 - AAX_IACFPageTable, [876](#)
 - AAX_IPageTable, [1056](#)
 - AAX_VPageTable, [1234](#)
- getMaskForSampleRate
 - AAX.h, [1318](#)
- GetMasterBypassParameter
 - AAX_CEffectParameters, [480](#)
 - AAX_IACFEffectParameters, [813](#)
- GetMaximumValue
 - AAX_CBinaryTaperDelegate< T >, [437](#)
 - AAX_CLinearTaperDelegate< T, RealPrecision >, [531](#)
 - AAX_CLogTaperDelegate< T, RealPrecision >, [536](#)
 - AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, [639](#)
 - AAX_CRangeTaperDelegate< T, RealPrecision >, [645](#)
 - AAX_CStateTaperDelegate< T >, [688](#)
 - AAX_ITaperDelegate< T >, [1109](#)
- GetMaxMinusMin
 - AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >, [742](#)
- GetMeterClipped
 - AAX_IACFController, [773](#)
 - AAX_IController, [979](#)

- AAX_VController, [1202](#)
- GetMeterCount
 - AAX_IACFController, [774](#)
 - AAX_IController, [979](#)
 - AAX_VController, [1202](#)
- GetMeterPeakValue
 - AAX_IACFController, [773](#)
 - AAX_IController, [978](#)
 - AAX_VController, [1201](#)
- GetMin
 - AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >, [742](#)
- GetMinimumValue
 - AAX_CBinaryTaperDelegate< T >, [437](#)
 - AAX_CLinearTaperDelegate< T, RealPrecision >, [530](#)
 - AAX_CLogTaperDelegate< T, RealPrecision >, [535](#)
 - AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, [639](#)
 - AAX_CRangeTaperDelegate< T, RealPrecision >, [645](#)
 - AAX_CStateTaperDelegate< T >, [688](#)
 - AAX_ITaperDelegate< T >, [1109](#)
- GetModifiers
 - AAX_IACFViewContainer, [932](#)
 - AAX_IViewContainer, [1130](#)
 - AAX_VViewContainer, [1269](#)
- GetNameVariationParameterIDAtIndex
 - AAX_IACFPageTable_V2, [880](#)
 - AAX_IPageTable, [1058](#)
 - AAX_VPageTable, [1235](#)
- GetNextMIDIPacket
 - AAX_IACFController, [774](#)
 - AAX_IController, [981](#)
 - AAX_VController, [1204](#)
- GetNodeBuffer
 - AAX_IMIDINode, [1049](#)
- GetNormalizedDefaultValue
 - AAX_CParameter< T >, [587](#)
 - AAX_CStatelessParameter, [669](#)
 - AAX_IPParameter, [1070](#)
- GetNormalizedValue
 - AAX_CParameter< T >, [588](#)
 - AAX_CStatelessParameter, [668](#)
 - AAX_IPParameter, [1070](#)
- GetNormalizedValueFromBool
 - AAX_CParameter< T >, [594, 609](#)
 - AAX_CStatelessParameter, [672](#)
 - AAX_IPParameter, [1073](#)
- GetNormalizedValueFromDouble
 - AAX_CParameter< T >, [595, 611](#)
 - AAX_CStatelessParameter, [674](#)
 - AAX_IPParameter, [1075](#)
- GetNormalizedValueFromFloat
 - AAX_CParameter< T >, [595, 610](#)
 - AAX_CStatelessParameter, [673](#)
 - AAX_IPParameter, [1074](#)
- GetNormalizedValueFromInt32
 - AAX_CParameter< T >, [594, 610](#)
 - AAX_CStatelessParameter, [673](#)
 - AAX_IPParameter, [1074](#)
- GetNormalizedValueFromStep
 - AAX_CParameter< T >, [589](#)
 - AAX_CStatelessParameter, [670](#)
 - AAX_IPParameter, [1072](#)
- GetNormalizedValueFromString
 - AAX_CParameter< T >, [596](#)
 - AAX_CStatelessParameter, [674](#)
 - AAX_IPParameter, [1075](#)
- GetNumberOfChanges
 - AAX_CEffectParameters, [495](#)
 - AAX_IACFEffEffectParameters, [828](#)
- GetNumberOfChunks
 - AAX_CEffectParameters, [493](#)
 - AAX_IACFEffEffectParameters, [826](#)
- GetNumberOfParameters
 - AAX_CEffectParameters, [480](#)
 - AAX_IACFEffEffectParameters, [813](#)
- GetNumberOfSteps
 - AAX_CParameter< T >, [589](#)
 - AAX_CStatelessParameter, [670](#)
 - AAX_IPParameter, [1071](#)
- GetNumBursts
 - AAX_IDma, [1010](#)
- GetNumMappedParameterIDs
 - AAX_IACFPageTable, [876](#)
 - AAX_IPageTable, [1055](#)
 - AAX_VPageTable, [1233](#)
- GetNumNameVariationsForParameter
 - AAX_IACFPageTable_V2, [880](#)
 - AAX_IPageTable, [1059](#)
 - AAX_VPageTable, [1236](#)
- GetNumOffsets
 - AAX_IDma, [1013](#)
- GetNumPages
 - AAX_IACFPageTable, [875](#)
 - AAX_IPageTable, [1053](#)
 - AAX_VPageTable, [1232](#)
- GetNumParametersWithNameVariations
 - AAX_IACFPageTable_V2, [879](#)
 - AAX_IPageTable, [1057](#)
 - AAX_VPageTable, [1235](#)
- GetOffsetTable
 - AAX_IDma, [1012](#)
- GetOrientation
 - AAX_CParameter< T >, [591](#)
 - AAX_CStatelessParameter, [683](#)
 - AAX_IPParameter, [1084](#)
- GetOutputRange
 - AAX_CHostProcessor, [522](#)
- GetOutputStemFormat
 - AAX_IACFController, [769](#)
 - AAX_IController, [973](#)
 - AAX_VController, [1193](#)
- GetParameter

- AAX_CEffectParameters, [484](#)
- AAX_CParameterManager, [619](#)
- AAX_IACFEffEffectParameters, [817](#)
- GetParameterByID
 - AAX_CParameterManager, [617](#), [618](#)
- GetParameterByName
 - AAX_CParameterManager, [618](#)
- GetParameterDefaultNormalizedValue
 - AAX_CEffectParameters, [482](#)
 - AAX_IACFEffEffectParameters, [815](#)
- GetParameterIDFromIndex
 - AAX_CEffectParameters, [485](#)
 - AAX_IACFEffEffectParameters, [818](#)
- GetParameterIndex
 - AAX_CEffectParameters, [484](#)
 - AAX_CParameterManager, [619](#)
 - AAX_IACFEffEffectParameters, [818](#)
- GetParameterIsAutomatable
 - AAX_CEffectParameters, [480](#)
 - AAX_IACFEffEffectParameters, [814](#)
- GetParameterName
 - AAX_CEffectParameters, [481](#)
 - AAX_IACFEffEffectParameters, [815](#)
- GetParameterNameOfLength
 - AAX_CEffectParameters, [481](#)
 - AAX_IACFEffEffectParameters, [815](#)
- GetParameterNameVariationAtIndex
 - AAX_IACFPageTable_V2, [881](#)
 - AAX_IPageTable, [1059](#)
 - AAX_VPageTable, [1236](#)
- GetParameterNameVariationOfLength
 - AAX_IACFPageTable_V2, [882](#)
 - AAX_IPageTable, [1060](#)
 - AAX_VPageTable, [1237](#)
- GetParameterNormalizedValue
 - AAX_CEffectParameters, [487](#)
 - AAX_IACFEffEffectParameters, [821](#)
- GetParameterNumberOfSteps
 - AAX_CEffectParameters, [481](#)
 - AAX_IACFEffEffectParameters, [814](#)
- GetParameterOrientation
 - AAX_CEffectParameters, [483](#)
 - AAX_IACFEffEffectParameters, [817](#)
- GetParameterStringFromValue
 - AAX_CEffectParameters, [486](#)
 - AAX_IACFEffEffectParameters, [820](#)
- GetParameterType
 - AAX_CEffectParameters, [483](#)
 - AAX_IACFEffEffectParameters, [816](#)
- GetParameterValueFromString
 - AAX_CEffectParameters, [485](#)
 - AAX_IACFEffEffectParameters, [819](#)
- GetParameterValueInfo
 - AAX_CEffectParameters, [485](#)
 - AAX_IACFEffEffectParameters, [819](#)
- GetParameterValueString
 - AAX_CEffectParameters, [486](#)
 - AAX_IACFEffEffectParameters, [820](#)
- GetPathToPlugInBundle
 - Other Extensions, [284](#)
- GetPlugInTargetPlatform
 - AAX_IACFController_V3, [782](#)
 - AAX_IController, [982](#)
 - AAX_VController, [1195](#)
- GetPointerProperty
 - AAX_IPropertyMap, [1096](#)
 - AAX_VPropertyMap, [1246](#)
- GetProgress
 - AAX_IACFTask, [908](#)
 - AAX_ITask, [1115](#)
 - AAX_VTask, [1255](#)
- GetProperty
 - AAX_IACFPropertyMap, [894](#)
 - AAX_IPropertyMap, [1096](#)
 - AAX_VPropertyMap, [1246](#)
- GetProperty64
 - AAX_IACFPropertyMap_V3, [900](#)
- GetPropertyWithIDArray
 - AAX_IACFPropertyMap_V2, [897](#)
 - AAX_IPropertyMap, [1099](#)
 - AAX_VPropertyMap, [1249](#)
- GetPtr
 - AAX_CPacket, [567](#), [568](#)
 - AAX_IACFViewContainer, [931](#)
 - AAX_IViewContainer, [1130](#)
 - AAX_VViewContainer, [1269](#)
- GetRPDFWithAmplitudeOne
 - AAX, [401](#)
- GetRPDFWithAmplitudeOneHalf
 - AAX, [401](#)
- GetSampleRate
 - AAX_IACFController, [768](#)
 - AAX_IController, [972](#)
 - AAX_VController, [1193](#)
- GetSessionDocument
 - AAX_CSessionDocumentClient, [653](#)
- GetSideChainInputNum
 - AAX_CHostProcessor, [524](#)
 - AAX_IACFHostProcessorDelegate, [861](#)
 - AAX_IHostProcessorDelegate, [1040](#)
 - AAX_VHostProcessorDelegate, [1224](#)
- GetSignalLatency
 - AAX_IACFController, [769](#)
 - AAX_IController, [973](#)
 - AAX_VController, [1194](#)
- GetSize
 - AAX_CPacket, [568](#)
 - AAX_Map, [1136](#)
- GetSrc
 - AAX_IDma, [1009](#)
- GetSrcEnd
 - AAX_CHostProcessor, [522](#)
- GetSrcStart
 - AAX_CHostProcessor, [522](#)
- GetStepValue
 - AAX_CParameter< T >, [589](#)

- AAX_CStatelessParameter, 670
- AAX_IParameter, 1071
- GetStepValueFromNormalizedValue
 - AAX_CParameter< T >, 590
 - AAX_CStatelessParameter, 670
 - AAX_IParameter, 1072
- GetStringFromNormalizedValue
 - AAX_CParameter< T >, 598
 - AAX_CStatelessParameter, 676, 677
 - AAX_IParameter, 1077, 1078
- GetTempoMap
 - AAX_ISessionDocument, 1101
 - AAX_VSessionDocument, 1251
- GetTicksPerQuarter
 - AAX_IACFTransport, 916
 - AAX_ITransport, 1124
 - AAX_VTransport, 1263
- GetTimeCodeInfo
 - AAX_IACFTransport_V2, 920
 - AAX_ITransport, 1125
 - AAX_VTransport, 1264
- GetTimelineSelectionEndPosition
 - AAX_IACFTransport_V4, 927
 - AAX_ITransport, 1127
 - AAX_VTransport, 1265
- GetTimelineSelectionStartPosition
 - AAX_IACFTransport_V2, 919
 - AAX_ITransport, 1125
 - AAX_VTransport, 1263
- Getting Started with AAX, 47
- GetTODLocation
 - AAX_IACFController, 770
 - AAX_IController, 974
 - AAX_VController, 1196
- GetTouchState
 - AAX_IACFAutomationDelegate, 746
 - AAX_IAutomationDelegate, 945
 - AAX_VAutomationDelegate, 1167
- GetTPDFWithAmplitudeOne
 - AAX, 402
- GetTransferSize
 - AAX_IDma, 1011
- GetTransport
 - AAX_IMIDINode, 1049
- GetType
 - AAX_CParameter< T >, 591
 - AAX_CStatelessParameter, 683
 - AAX_IACFTask, 907
 - AAX_IACFViewContainer, 931
 - AAX_IParameter, 1083
 - AAX_ITask, 1114
 - AAX_IViewContainer, 1130
 - AAX_VTask, 1254
 - AAX_VViewContainer, 1268
- GetUnknown
 - AAX_VAutomationDelegate, 1165
- GetUpperBoundIndex
 - AAX_Map, 1135
- GetValue
 - AAX_CParameter< T >, 606
- GetValueAsBool
 - AAX_CParameter< T >, 601
 - AAX_CParameterValue< T >, 626, 628
 - AAX_CStatelessParameter, 678
 - AAX_IParameter, 1079
 - AAX_IParameterValue, 1087
- GetValueAsDouble
 - AAX_CParameter< T >, 602
 - AAX_CParameterValue< T >, 627, 629
 - AAX_CStatelessParameter, 679
 - AAX_IParameter, 1080
 - AAX_IParameterValue, 1088
- GetValueAsFloat
 - AAX_CParameter< T >, 602
 - AAX_CParameterValue< T >, 627, 629
 - AAX_CStatelessParameter, 679
 - AAX_IParameter, 1079
 - AAX_IParameterValue, 1088
- GetValueAsInt32
 - AAX_CParameter< T >, 601
 - AAX_CParameterValue< T >, 626, 629
 - AAX_CStatelessParameter, 679
 - AAX_IParameter, 1079
 - AAX_IParameterValue, 1087
- GetValueAsString
 - AAX_CParameter< T >, 602, 607
 - AAX_CParameterValue< T >, 628, 630
 - AAX_CStatelessParameter, 680
 - AAX_IParameter, 1080
 - AAX_IParameterValue, 1088
- GetValueString
 - AAX_CParameter< T >, 593
 - AAX_CStatelessParameter, 671
 - AAX_IParameter, 1072, 1073
- GetViewContainer
 - AAX_CEffectGUI, 468
- GetViewContainerPtr
 - AAX_CEffectGUI, 468
- GetViewContainerType
 - AAX_CEffectGUI, 468
- GetViewSize
 - AAX_CEffectGUI, 464
 - AAX_IACFEEffectGUI, 803
- GetX
 - AAX_Map, 1136
- GetY
 - AAX_Map, 1136
- GUI Extensions, 282
- GUI interface, 74
- HandleAssertFailure
 - AAX_CHostServices, 526
 - AAX_IACFHostServices_V3, 872
 - AAX_IHostServices, 1042
 - AAX_VHostServices, 1226
- HandleMultipleParametersMouseDown
 - AAX_IACFViewContainer_V2, 936

- AAX_IViewContainer, [1133](#)
- AAX_VViewContainer, [1272](#)
- HandleMultipleParametersMouseDown
 - AAX_IACFViewContainer_V2, [936](#)
 - AAX_IViewContainer, [1133](#)
 - AAX_VViewContainer, [1272](#)
- HandleMultipleParametersMouseUp
 - AAX_IACFViewContainer_V2, [937](#)
 - AAX_IViewContainer, [1134](#)
 - AAX_VViewContainer, [1273](#)
- HandleParameterMouseDown
 - AAX_IACFViewContainer, [932](#)
 - AAX_IViewContainer, [1131](#)
 - AAX_VViewContainer, [1270](#)
- HandleParameterMouseDrag
 - AAX_IACFViewContainer, [933](#)
 - AAX_IViewContainer, [1131](#)
 - AAX_VViewContainer, [1270](#)
- HandleParameterMouseEnter
 - AAX_IACFViewContainer_V3, [940](#)
 - AAX_IViewContainer, [1132](#)
 - AAX_VViewContainer, [1271](#)
- HandleParameterMouseExit
 - AAX_IACFViewContainer_V3, [940](#)
 - AAX_IViewContainer, [1132](#)
 - AAX_VViewContainer, [1271](#)
- HandleParameterMouseUp
 - AAX_IACFViewContainer, [933](#)
 - AAX_IViewContainer, [1132](#)
 - AAX_VViewContainer, [1271](#)
- HDX DSP Guide, [178](#)
- HEADER_SIZE
 - AAX_ChunkDataParserDefs, [410](#)
- height
 - AAX_Rect, [1139](#)
- horz
 - AAX_Point, [1138](#)
- Host Support, [295](#)
- HostDefinition
 - AAX_ICollection, [950](#)
 - AAX_VCollection, [1174](#)
 - AAX_VDescriptionHost, [1212](#)
- HostProcessorDelegate
 - AAX_CHostProcessor, [524](#), [525](#)
- Hybrid Processing architecture, [84](#)
 - GetHybridSignalLatency, [86](#)
 - RenderAudio_Hybrid, [86](#)
- IACFDefinition, [1279](#)
 - CopyAttribute, [1281](#)
 - DefineAttribute, [1280](#)
 - GetAttributeInfo, [1281](#)
- IACFUnknown, [1282](#)
 - AddRef, [1283](#)
 - QueryInterface, [1283](#)
 - Release, [1284](#)
- ID
 - AAX_IFeatureInfo, [1034](#)
 - AAX_VFeatureInfo, [1221](#)
- Identifier
 - AAX_CParameter< T >, [585](#)
 - AAX_CParameterValue< T >, [626](#)
 - AAX_CStatelessParameter, [662](#)
 - AAX_IParameter, [1066](#)
 - AAX_IParameterValue, [1086](#)
- IID_IAAXAutomationDelegateV1
 - AAX_UIDs.h, [1658](#)
- IID_IAAXCollectionV1
 - AAX_UIDs.h, [1655](#)
- IID_IAAXComponentDescriptorV1
 - AAX_UIDs.h, [1656](#)
- IID_IAAXComponentDescriptorV2
 - AAX_UIDs.h, [1656](#)
- IID_IAAXComponentDescriptorV3
 - AAX_UIDs.h, [1657](#)
- IID_IAAXControllerV1
 - AAX_UIDs.h, [1659](#)
- IID_IAAXControllerV2
 - AAX_UIDs.h, [1659](#)
- IID_IAAXControllerV3
 - AAX_UIDs.h, [1659](#)
- IID_IAAXDataBufferV1
 - AAX_UIDs.h, [1667](#)
- IID_IAAXDescriptionHostV1
 - AAX_UIDs.h, [1662](#)
- IID_IAAXEffectDescriptorV1
 - AAX_UIDs.h, [1656](#)
- IID_IAAXEffectDescriptorV2
 - AAX_UIDs.h, [1656](#)
- IID_IAAXEffectDirectDataV1
 - AAX_UIDs.h, [1665](#)
- IID_IAAXEffectDirectDataV2
 - AAX_UIDs.h, [1666](#)
- IID_IAAXEffectGUIV1
 - AAX_UIDs.h, [1665](#)
- IID_IAAXEffectParametersV1
 - AAX_UIDs.h, [1664](#)
- IID_IAAXEffectParametersV2
 - AAX_UIDs.h, [1664](#)
- IID_IAAXEffectParametersV3
 - AAX_UIDs.h, [1664](#)
- IID_IAAXEffectParametersV4
 - AAX_UIDs.h, [1664](#)
- IID_IAAXFeatureInfoV1
 - AAX_UIDs.h, [1663](#)
- IID_IAAXHostProcessorDelegateV1
 - AAX_UIDs.h, [1658](#)
- IID_IAAXHostProcessorDelegateV2
 - AAX_UIDs.h, [1658](#)
- IID_IAAXHostProcessorDelegateV3
 - AAX_UIDs.h, [1658](#)
- IID_IAAXHostProcessorV1
 - AAX_UIDs.h, [1665](#)
- IID_IAAXHostProcessorV2
 - AAX_UIDs.h, [1665](#)
- IID_IAAXHostServicesV1
 - AAX_UIDs.h, [1655](#)

- IID_IAAXHostServicesV2
 - AAX_UIDs.h, [1655](#)
- IID_IAAXHostServicesV3
 - AAX_UIDs.h, [1655](#)
- IID_IAAXPageTableController
 - AAX_UIDs.h, [1659](#)
- IID_IAAXPageTableControllerV2
 - AAX_UIDs.h, [1659](#)
- IID_IAAXPageTableV1
 - AAX_UIDs.h, [1662](#)
- IID_IAAXPageTableV2
 - AAX_UIDs.h, [1662](#)
- IID_IAAXPrivateDataAccessV1
 - AAX_UIDs.h, [1660](#)
- IID_IAAXPropertyMapV1
 - AAX_UIDs.h, [1657](#)
- IID_IAAXPropertyMapV2
 - AAX_UIDs.h, [1657](#)
- IID_IAAXPropertyMapV3
 - AAX_UIDs.h, [1657](#)
- IID_IAAXSessionDocumentClientV1
 - AAX_UIDs.h, [1666](#)
- IID_IAAXSessionDocumentV1
 - AAX_UIDs.h, [1663](#)
- IID_IAAXTaskAgentV1
 - AAX_UIDs.h, [1666](#)
- IID_IAAXTaskV1
 - AAX_UIDs.h, [1663](#)
- IID_IAAXTransportControlV1
 - AAX_UIDs.h, [1662](#)
- IID_IAAXTransportV1
 - AAX_UIDs.h, [1661](#)
- IID_IAAXTransportV2
 - AAX_UIDs.h, [1661](#)
- IID_IAAXTransportV3
 - AAX_UIDs.h, [1661](#)
- IID_IAAXTransportV4
 - AAX_UIDs.h, [1661](#)
- IID_IAAXViewContainerV1
 - AAX_UIDs.h, [1660](#)
- IID_IAAXViewContainerV2
 - AAX_UIDs.h, [1660](#)
- IID_IAAXViewContainerV3
 - AAX_UIDs.h, [1660](#)
- Initialize
 - AAX_CEffectDirectData, [455](#)
 - AAX_CEffectGUI, [462](#)
 - AAX_CEffectParameters, [478](#)
 - AAX_CHostProcessor, [516](#)
 - AAX_CPacketDispatcher, [569](#)
 - AAX_CParameterManager, [616](#)
 - AAX_CSessionDocumentClient, [650](#)
 - AAX_CTaskAgent, [726](#)
 - AAX_IACFEEffectDirectData, [796](#)
 - AAX_IACFEEffectGUI, [802](#)
 - AAX_IACFEEffectParameters, [812](#)
 - AAX_IACFHostProcessor, [852](#)
 - AAX_IACFSessionDocumentClient, [904](#)
 - AAX_IACFTaskAgent, [910](#)
- Initialize_PrivateDataAccess
 - AAX_CEffectDirectData, [457](#)
- InitOutputBounds
 - AAX_CHostProcessor, [517](#)
 - AAX_IACFHostProcessor, [852](#)
- Insert
 - AAX_CString, [698](#), [699](#)
- InsertHex
 - AAX_CString, [699](#)
- InsertNumber
 - AAX_CString, [699](#)
- InsertPage
 - AAX_IACFPageTable, [875](#)
 - AAX_IPageTable, [1055](#)
 - AAX_VPageTable, [1232](#)
- Int32ToNormalized
 - AAX_CEffectParameters.h, [1364](#)
- IsAccentedClick
 - AAX, [378](#)
- IsAllNotesOff
 - AAX, [378](#)
- IsASCII
 - AAX, [384](#)
- IsAvidNotification
 - AAX, [390](#)
- IsClick
 - AAX, [379](#)
- IsDirty
 - AAX_CPacket, [568](#)
- IsEffectIDEqual
 - AAX, [390](#)
- IsEmpty
 - AAX_CChunkDataParser, [444](#)
- IsFourCharASCII
 - AAX, [385](#)
- IsMetronomeEnabled
 - AAX_IACFTransport_V2, [921](#)
 - AAX_ITransport, [1126](#)
 - AAX_VTransport, [1264](#)
- IsNoteOff
 - AAX, [378](#)
- IsNoteOn
 - AAX, [378](#)
- IsParameterIDEqual
 - AAX, [390](#)
- IsParameterLinkReady
 - AAX_CEffectParameters, [503](#)
- IsParameterTouched
 - AAX_CEffectParameters, [503](#)
- IsSupported
 - AAX_VPageTable, [1240](#)
- IsTransferComplete
 - AAX_IDma, [1007](#)
- IsTransportPlaying
 - AAX_IACFTransport, [914](#)
 - AAX_ITransport, [1121](#)
 - AAX_VTransport, [1260](#)

- IsUnaccentedClick
 - AAX, [379](#)
- k32BitAbsMax
 - AAX_CommonConversions.h, [1380](#)
- k32BitNegMax
 - AAX_CommonConversions.h, [1381](#)
- k32BitPosMax
 - AAX_CommonConversions.h, [1380](#)
- k56kFloatNegMax
 - AAX_CommonConversions.h, [1382](#)
- k56kFloatPosMax
 - AAX_CommonConversions.h, [1382](#)
- k56kFracAbsMax
 - AAX_CommonConversions.h, [1381](#)
- k56kFracHalf
 - AAX_CommonConversions.h, [1381](#)
- k56kFracNegMax
 - AAX_CommonConversions.h, [1381](#)
- k56kFracNegOne
 - AAX_CommonConversions.h, [1381](#)
- k56kFracPosMax
 - AAX_CommonConversions.h, [1381](#)
- k56kFracZero
 - AAX_CommonConversions.h, [1381](#)
- kAAX_DataBufferType_TempoBreakpointArray
 - AAX_SessionDocumentTypes.h, [1630](#)
- kAAX_eTargetPlatform_Count
 - AAX_Enums.h, [1473](#)
- kAAX_eTargetPlatform_External
 - AAX_Enums.h, [1473](#)
- kAAX_eTargetPlatform_Native
 - AAX_Enums.h, [1473](#)
- kAAX_eTargetPlatform_None
 - AAX_Enums.h, [1473](#)
- kAAX_eTargetPlatform_TI
 - AAX_Enums.h, [1473](#)
- kAAX_ParameterIdentifierMaxSize
 - AAX.h, [1318](#)
- kAAX_ProcPtrID_Create_EffectDirectData
 - AAX_Callbacks.h, [1342](#)
- kAAX_ProcPtrID_Create_EffectGUI
 - AAX_Callbacks.h, [1342](#)
- kAAX_ProcPtrID_Create_EffectParameters
 - AAX_Callbacks.h, [1342](#)
- kAAX_ProcPtrID_Create_HostProcessor
 - AAX_Callbacks.h, [1342](#)
- kAAX_ProcPtrID_Create_SessionDocumentClient
 - AAX_Callbacks.h, [1342](#)
- kAAX_ProcPtrID_Create_TaskAgent
 - AAX_Callbacks.h, [1342](#)
- kAAX_Trace_Priority_Critical
 - AAX_Assert.h, [1325](#)
- kAAX_Trace_Priority_High
 - AAX_Assert.h, [1325](#)
- kAAX_Trace_Priority_Low
 - AAX_Assert.h, [1325](#)
- kAAX_Trace_Priority_Lowest
 - AAX_Assert.h, [1325](#)
- kAAX_Trace_Priority_None
 - AAX_Assert.h, [1325](#)
- kAAX_Trace_Priority_Normal
 - AAX_Assert.h, [1325](#)
- kInvalidIndex
 - AAX_CString, [706](#)
- kMaxAdditionalMIDINodes
 - AAX_CMonolithicParameters.h, [1299](#)
- kMaxAuxOutputStems
 - AAX_CMonolithicParameters.h, [1299](#)
- kMaxStringLength
 - AAX_CString, [706](#)
- kNeg144DB
 - AAX_CommonConversions.h, [1382](#)
- kNeg144Gain
 - AAX_CommonConversions.h, [1382](#)
- Known Issues, [302](#)
- kOneOver56kFracAbsMax
 - AAX_CommonConversions.h, [1382](#)
- kPowExtent
 - AAX, [406](#)
- kPowTableSize
 - AAX, [406](#)
- kSynchronizedParameterQueueSize
 - AAX_CMonolithicParameters.h, [1299](#)
- LastError
 - AAX_CheckedResult, [511](#)
- LastFailure
 - AAX_AggregateResult, [415](#)
- left
 - AAX_Rect, [1139](#)
- Length
 - AAX_CString, [693](#)
 - AAX_IMIDIMessageInfoDelegate, [1044](#)
 - AAX_IString, [1105](#)
- Line
 - AAX::Exception::Any, [1276](#)
- Linked parameter update sequences, [136](#)
- Linked parameters, [131](#)
- LoadChunk
 - AAX_CChunkDataParser, [445](#)
- Lock
 - AAX_CMutex, [559](#)
- LogDoubleToLongControl
 - AAX_SliderConversions.h, [1634](#)
- LONG_STRING_IDENTIFIER
 - AAX_ChunkDataParserDefs, [408](#)
- LONG_TYPE
 - AAX_ChunkDataParserDefs, [408](#)
- LongControlToDouble
 - AAX_SliderConversions.h, [1633](#)
- LongControlToDoubleNonlinear
 - AAX_SliderConversions.h, [1633](#)
- LongControlToLogDouble
 - AAX_SliderConversions.h, [1634](#)
- LongControlToNewRange
 - AAX_SliderConversions.h, [1632](#)
- LongToDouble

- AAX_CommonConversions.h, [1377](#)
- LongToLongControl
 - AAX_SliderConversions.h, [1633](#)
- mAdditionalInputMIDINodes
 - AAX_SInstrumentRenderInfo, [1144](#)
- MapParameterID
 - AAX_IACFPageTable, [877](#)
 - AAX_IPageTable, [1057](#)
 - AAX_VPageTable, [1234](#)
- Mask
 - AAX_IMIDIMessageInfoDelegate, [1044](#)
- mAudioInputs
 - AAX_SHybridRenderInfo, [1140](#)
 - AAX_SInstrumentRenderInfo, [1143](#)
- mAudioOutputs
 - AAX_SHybridRenderInfo, [1140](#)
 - AAX_SInstrumentRenderInfo, [1143](#)
- mAudiosuiteID
 - AAX_SInstrumentSetupInfo, [1153](#)
- mAutomatable
 - AAX_CParameter< T >, [613](#)
- mAutomationDelegate
 - AAX_CParameter< T >, [614](#)
 - AAX_CParameterManager, [620](#)
 - AAX_CStatelessParameter, [685](#)
- mAuxOutputStemFormats
 - AAX_SInstrumentSetupInfo, [1150](#)
- mAuxOutputStemNames
 - AAX_SInstrumentSetupInfo, [1150](#)
- Max
 - AAX, [395](#)
- MAX_NAME_LENGTH
 - AAX_ChunkDataParserDefs, [410](#)
- MAX_STRINGDATA_LENGTH
 - AAX_ChunkDataParserDefs, [409](#)
- MaxLength
 - AAX_CString, [694](#)
 - AAX_IString, [1105](#)
- mBuffer
 - AAX_CMidiStream, [541](#)
- mBufferSize
 - AAX_CMidiStream, [540](#)
- mCanBypass
 - AAX_SInstrumentSetupInfo, [1152](#)
- mChunkData
 - AAX_CChunkDataParser, [446](#)
- mChunkParser
 - AAX_CEffectParameters, [505](#)
- mChunkSize
 - AAX_CEffectParameters, [505](#)
- mChunkVersion
 - AAX_CChunkDataParser, [446](#)
- mClock
 - AAX_SHybridRenderInfo, [1141](#)
 - AAX_SInstrumentRenderInfo, [1144](#)
- mControlType
 - AAX_CParameter< T >, [613](#)
- mCurrentStateNum
 - AAX_SInstrumentRenderInfo, [1145](#)
- mData
 - AAX_CMidiPacket, [539](#)
- mDataName
 - AAX_CChunkDataParser::DataValue, [1278](#)
- mDataType
 - AAX_CChunkDataParser::DataValue, [1278](#)
- mDataValues
 - AAX_CChunkDataParser, [446](#)
- mDefaultValue
 - AAX_CParameter< T >, [614](#)
- mDisplayDelegate
 - AAX_CParameter< T >, [614](#)
- Media Composer Guide, [170](#)
- mFilteredParameters
 - AAX_CEffectParameters, [506](#)
- mGlobalMIDIEventMask
 - AAX_SInstrumentSetupInfo, [1147](#)
- mGlobalMIDI nodeName
 - AAX_SInstrumentSetupInfo, [1147](#)
- mGlobalNode
 - AAX_SInstrumentRenderInfo, [1144](#)
- mHybridInputStemFormat
 - AAX_SInstrumentSetupInfo, [1150](#)
- mHybridOutputStemFormat
 - AAX_SInstrumentSetupInfo, [1151](#)
- mID
 - AAX_CStatelessParameter, [685](#)
- MIDI, [87](#)
- Min
 - AAX, [396](#)
- MinMax
 - AAX, [395](#)
- mInputMIDIChannelMask
 - AAX_SInstrumentSetupInfo, [1148](#)
- mInputMIDI nodeName
 - AAX_SInstrumentSetupInfo, [1148](#)
- mInputNode
 - AAX_SInstrumentRenderInfo, [1144](#)
- mInputStemFormat
 - AAX_SInstrumentSetupInfo, [1151](#)
- mIntValue
 - AAX_CChunkDataParser::DataValue, [1278](#)
- mInverseStringMap
 - AAX_CStringDisplayDelegate< T >, [722](#)
- mIsImmediate
 - AAX_CMidiPacket, [539](#)
- mIsLoopEnabled
 - AAX_TransportStateInfo_V1, [1163](#)
- mIsRecordEnabled
 - AAX_TransportStateInfo_V1, [1162](#)
- mIsRecording
 - AAX_TransportStateInfo_V1, [1163](#)
- mLastFoundIndex
 - AAX_CChunkDataParser, [445](#)
- mLength
 - AAX_CMidiPacket, [539](#)
- mManufacturerID

- AAX_SInstrumentSetupInfo, [1152](#)
- AAX_SPlugInIdentifierTriad, [1158](#)
- mMeterIDs
 - AAX_SInstrumentSetupInfo, [1149](#)
- mMeters
 - AAX_SInstrumentRenderInfo, [1145](#)
- mMonolithicParametersPtr
 - AAX_SInstrumentPrivateData, [1142](#)
- mMultiMonoSupport
 - AAX_SInstrumentSetupInfo, [1153](#)
- mNames
 - AAX_CParameter< T >, [613](#)
 - AAX_CStatelessParameter, [685](#)
- mNeedNotify
 - AAX_CParameter< T >, [614](#)
- mNeedsGlobalMIDI
 - AAX_SInstrumentSetupInfo, [1147](#)
- mNeedsInputMIDI
 - AAX_SInstrumentSetupInfo, [1148](#)
- mNeedsTransport
 - AAX_SInstrumentSetupInfo, [1149](#)
- mNumAdditionalInputMIDINodes
 - AAX_SInstrumentSetupInfo, [1148](#)
- mNumAudioInputs
 - AAX_SHybridRenderInfo, [1140](#)
- mNumAudioOutputs
 - AAX_SHybridRenderInfo, [1141](#)
- mNumAuxOutputStems
 - AAX_SInstrumentSetupInfo, [1150](#)
- mNumChunkedParameters
 - AAX_CEffectParameters, [505](#)
- mNumMeters
 - AAX_SInstrumentSetupInfo, [1149](#)
- mNumPlugInChanges
 - AAX_CEffectParameters, [505](#)
- mNumSamples
 - AAX_SHybridRenderInfo, [1141](#)
 - AAX_SInstrumentRenderInfo, [1143](#)
- mNumSteps
 - AAX_CParameter< T >, [613](#)
- Monolithic VIs and Effects, [283](#)
- mOrientation
 - AAX_CParameter< T >, [614](#)
- mOutputStemFormat
 - AAX_SInstrumentSetupInfo, [1151](#)
- mPacketDispatcher
 - AAX_CEffectParameters, [506](#)
- mParameterManager
 - AAX_CEffectParameters, [506](#)
- mParameters
 - AAX_CParameterManager, [621](#)
- mParametersMap
 - AAX_CParameterManager, [621](#)
- mPlugInID
 - AAX_SPlugInIdentifierTriad, [1159](#)
- mPluginID
 - AAX_SInstrumentSetupInfo, [1152](#)
- mPrivateData
 - AAX_SInstrumentRenderInfo, [1144](#)
- mProductID
 - AAX_SInstrumentSetupInfo, [1152](#)
 - AAX_SPlugInIdentifierTriad, [1159](#)
- mRecordMode
 - AAX_TransportStateInfo_V1, [1162](#)
- mSampleLocation
 - AAX_CTempoBreakpoint, [728](#)
- mString
 - AAX_CString, [706](#)
- mStringMap
 - AAX_CStringDisplayDelegate< T >, [722](#)
- mStringValue
 - AAX_CChunkDataParser::DataValue, [1278](#)
- mTaperDelegate
 - AAX_CParameter< T >, [614](#)
- mTimestamp
 - AAX_CMidiPacket, [539](#)
- mTransportMIDINodeName
 - AAX_SInstrumentSetupInfo, [1149](#)
- mTransportNode
 - AAX_SInstrumentRenderInfo, [1144](#)
- mTransportState
 - AAX_TransportStateInfo_V1, [1162](#)
- mUnitString
 - AAX_CUnitDisplayDelegateDecorator< T >, [734](#)
- mUseHostGeneratedGUI
 - AAX_SInstrumentSetupInfo, [1151](#)
- mValue
 - AAX_CParameter< T >, [614](#)
 - AAX_CTempoBreakpoint, [728](#)
- mValueString
 - AAX_CStatelessParameter, [685](#)
- Name
 - AAX_CParameter< T >, [586](#)
 - AAX_CStatelessParameter, [664](#)
 - AAX_IPParameter, [1067](#)
- NAME_NOT_FOUND
 - AAX_ChunkDataParserDefs, [410](#)
- NewComponentDescriptor
 - AAX_IEffectDescriptor, [1016](#)
 - AAX_VEffectDescriptor, [1215](#)
- NewDescriptor
 - AAX_ICollection, [947](#)
 - AAX_VCollection, [1170](#)
- NewPropertyMap
 - AAX_ICollection, [949](#)
 - AAX_IComponentDescriptor, [963](#)
 - AAX_IEffectDescriptor, [1018](#)
 - AAX_VCollection, [1173](#)
 - AAX_VComponentDescriptor, [1186](#)
 - AAX_VEffectDescriptor, [1217](#)
- None
 - Task agent interface, [104](#)
- NormalizedToInt32
 - AAX_CEffectParameters.h, [1364](#)
- NormalizedToReal
 - AAX_CBinaryTaperDelegate< T >, [438](#)

- AAX_CLinearTaperDelegate< T, RealPrecision >, 531
- AAX_CLogTaperDelegate< T, RealPrecision >, 536
- AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, 640
- AAX_CRangeTaperDelegate< T, RealPrecision >, 646
- AAX_CStateTaperDelegate< T >, 689
- AAX_ITaperDelegate< T >, 1110
- NotificationReceived
 - AAX_CEffectDirectData, 456
 - AAX_CEffectGUI, 462
 - AAX_CEffectParameters, 479
 - AAX_CSessionDocumentClient, 651
 - AAX_IACFEffctDirectData_V2, 798
 - AAX_IACFEffctGUI, 802
 - AAX_IACFEffctParameters, 812
 - AAX_IACFSessionDocumentClient, 905
- NumAttempted
 - AAX_AggregateResult, 416
- NumFailed
 - AAX_AggregateResult, 415
- NumParameters
 - AAX_CParameterManager, 617
- NumSucceeded
 - AAX_AggregateResult, 415
- Offline processing interface, 83
- operator AAX_Result
 - AAX_AggregateResult, 413
 - AAX_CheckedResult, 510
- operator!=
 - AAX_CString, 704
 - AAX_GUITypes.h, 1516, 1517
 - AAX_TransportTypes.h, 1649
- operator<
 - AAX_CString, 704
 - AAX_GUITypes.h, 1516
- operator<<
 - AAX_CString, 706
- operator<=
 - AAX_GUITypes.h, 1517
- operator>
 - AAX_CString, 705
 - AAX_GUITypes.h, 1517
- operator>>
 - AAX_CString, 706
- operator>=
 - AAX_GUITypes.h, 1517
- operator+
 - AAX_CString.h, 1417
- operator+=
 - AAX_CString, 705
- operator=
 - AAX::Exception::Any, 1275
 - AAX_AggregateResult, 413
 - AAX_CArrayDataBuffer< D >, 419
 - AAX_CArrayDataBufferOfType< T, D >, 423
 - AAX_CEffectParameters, 478
 - AAX_CheckedResult, 509
 - AAX_CRangeTaperDelegate< T, RealPrecision >, 644
 - AAX_CString, 695, 696
 - AAX_CStringDataBuffer, 712, 713
 - AAX_CStringDataBufferOfType< T >, 717
 - AAX_IString, 1106
- operator==
 - AAX_CString, 703, 704
 - AAX_GUITypes.h, 1516, 1517
 - AAX_TransportTypes.h, 1649
- operator[]
 - AAX_CString, 705
- operator |=
 - AAX_CheckedResult, 510
- Other Extensions, 284
 - AsStringMIDIStream_Debug, 284
 - GetPathToPlugInBundle, 284
- override
 - AAX_IEffectDirectData, 1023
 - AAX_IEffectGUI, 1026
 - AAX_IEffectParameters, 1033
 - AAX_IHostProcessor, 1038
 - AAX_ISessionDocumentClient, 1104
- Page Table Guide, 224
- PageTableParameterMappingsAreEqual
 - AAX, 380
- PageTableParameterNameVariationsAreEqual
 - AAX, 380
- PageTablesAreEqual
 - AAX, 381
- Parameter automation, 113
- Parameter Manager, 105
- Parameter update timing, 116
- Parameter updates, 115
- ParameterNameChanged
 - AAX_IAutomationDelegate, 945
 - AAX_VAutomationDelegate, 1168
- ParameterUpdated
 - AAX_CEffectGUI, 465
 - AAX_IACFEffctGUI, 804
- Peek
 - AAX_CAtomicQueue< T, S >, 428
 - AAX_IPointerQueue< T >, 1092
- Plug-in meters, 90
- Plug-in type conversion, 140
- PolyEval
 - AAX, 396
- Pop
 - AAX_CAtomicQueue< T, S >, 428
 - AAX_IPointerQueue< T >, 1091
- PostAnalyze
 - AAX_CHostProcessor, 520
 - AAX_IACFHostProcessor, 856
- PostCurrentValue
 - AAX_IACFAutomationDelegate, 745
 - AAX_IAutomationDelegate, 944

- AAX_VAutomationDelegate, [1166](#)
- PostMIDIPacket
 - AAX_IMIDINode, [1049](#)
- PostPacket
 - AAX_IACFController, [772](#)
 - AAX_IController, [976](#)
 - AAX_VController, [1199](#)
- PostReleaseRequest
 - AAX_IACFAutomationDelegate, [746](#)
 - AAX_IAutomationDelegate, [944](#)
 - AAX_VAutomationDelegate, [1166](#)
- PostRender
 - AAX_CHostProcessor, [519](#)
 - AAX_IACFHostProcessor, [855](#)
- PostRequest
 - AAX_IDma, [1007](#)
- PostSetValueRequest
 - AAX_IACFAutomationDelegate, [745](#)
 - AAX_IAutomationDelegate, [943](#)
 - AAX_VAutomationDelegate, [1166](#)
- PostTouchRequest
 - AAX_IACFAutomationDelegate, [745](#)
 - AAX_IAutomationDelegate, [944](#)
 - AAX_VAutomationDelegate, [1167](#)
- PreAnalyze
 - AAX_CHostProcessor, [520](#)
 - AAX_IACFHostProcessor, [856](#)
- PreRender
 - AAX_CHostProcessor, [519](#)
 - AAX_IACFHostProcessor, [854](#)
- Primary
 - AAX_CStringAbbreviations, [707](#)
- Pro Tools Guide, [151](#)
- pt2Object
 - AAX_CPacketHandler< TWorker >, [574](#)
- Push
 - AAX_CAtomicQueue< T, S >, [427](#)
 - AAX_IPointerQueue< T >, [1091](#)
- QueryInterface
 - IACFUnknown, [1283](#)
- ReadMe.doxygen, [1296](#)
- ReadPortDirect
 - AAX_IACFPrivateDataAccess, [892](#)
 - AAX_IPrivateDataAccess, [1093](#)
 - AAX_VPrivateDataAccess, [1242](#)
- Real-time algorithm callback, [66](#)
- Real-time performance, [112](#)
- RealToNormalized
 - AAX_CBinaryTaperDelegate< T >, [438](#)
 - AAX_CLinearTaperDelegate< T, RealPrecision >, [532](#)
 - AAX_CLogTaperDelegate< T, RealPrecision >, [537](#)
 - AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, [640](#)
 - AAX_CRangeTaperDelegate< T, RealPrecision >, [646](#)
 - AAX_CStateTaperDelegate< T >, [689](#)
 - AAX_ITaperDelegate< T >, [1110](#)
- ReceiveTask
 - AAX_CTaskAgent, [727](#)
- RegisterPacket
 - AAX_CPacketDispatcher, [569](#), [570](#)
- RegisterParameter
 - AAX_IACFAutomationDelegate, [744](#)
 - AAX_IAutomationDelegate, [942](#)
 - AAX_VAutomationDelegate, [1165](#)
- Release
 - AAX_CParameter< T >, [600](#)
 - AAX_CStatelessParameter, [667](#)
 - AAX_IPParameter, [1070](#)
 - IACFUnknown, [1284](#)
- ReleaseParameter
 - AAX_CEffectParameters, [489](#)
 - AAX_IACFEffEffectParameters, [823](#)
- RemoveAllParameters
 - AAX_CParameterManager, [617](#)
- RemovePage
 - AAX_IACFPageTable, [875](#)
 - AAX_IPageTable, [1055](#)
 - AAX_VPageTable, [1232](#)
- RemoveParameter
 - AAX_CParameterManager, [620](#)
- RemoveParameterByID
 - AAX_CParameterManager, [617](#)
- RemoveProperty
 - AAX_IACFPropertyMap, [895](#)
 - AAX_IPropertyMap, [1098](#)
 - AAX_VPropertyMap, [1248](#)
- RenderAudio
 - AAX_CHostProcessor, [518](#)
 - AAX_CMonolithicParameters, [550](#)
 - AAX_IACFHostProcessor, [854](#)
- RenderAudio_Hybrid
 - AAX_CEffectParameters, [501](#)
 - Hybrid Processing architecture, [86](#)
- Replace
 - AAX_CString, [699](#)
- ReplaceDouble
 - AAX_CChunkDataParser, [444](#)
- RequestTransportStart
 - AAX_IACFTransportControl, [929](#)
 - AAX_ITransport, [1127](#)
 - AAX_VTransport, [1266](#)
- RequestTransportStop
 - AAX_IACFTransportControl, [929](#)
 - AAX_ITransport, [1127](#)
 - AAX_VTransport, [1266](#)
- Reset
 - SArray< T >, [1288](#)
- ResetAcceptedResults
 - AAX_CheckedResult, [509](#)
- ResetFieldData
 - AAX_CEffectParameters, [492](#)
 - AAX_CMonolithicParameters, [554](#)

- AAX_IACFEffEffectParameters, [825](#)
- Result
 - AAX::Exception::ResultError, [1286](#)
- ResultError
 - AAX::Exception::ResultError, [1286](#)
- Round
 - AAX_CLinearTaperDelegate< T, RealPrecision >, [532](#)
 - AAX_CLogTaperDelegate< T, RealPrecision >, [538](#)
 - AAX_CPieceWiseLinearTaperDelegate< T, RealPrecision >, [640](#)
 - AAX_CRangeTaperDelegate< T, RealPrecision >, [646](#)
- SafeLog
 - AAX, [389](#)
- SafeLogf
 - AAX, [389](#)
- sampleRateInMask
 - AAX.h, [1317](#)
- SAutoArray
 - SAutoArray< T >, [1287](#)
- SAutoArray< T >, [1287](#)
 - ~SAutoArray, [1287](#)
 - Get, [1288](#)
 - Reset, [1288](#)
 - SAutoArray, [1287](#)
- SendNotification
 - AAX_IACFController_V2, [777](#)
 - AAX_IController, [977](#), [978](#)
 - AAX_VController, [1200](#)
- SessionDocumentChanged
 - AAX_CSessionDocumentClient, [652](#)
- SessionDocumentWillChange
 - AAX_CSessionDocumentClient, [651](#)
- Set
 - AAX_CHostServices, [526](#)
 - AAX_CParameterValue< T >, [625](#)
 - AAX_CString, [695](#)
 - AAX_IString, [1106](#)
- SetAutomationDelegate
 - AAX_CParameter< T >, [599](#)
 - AAX_CStatelessParameter, [666](#)
 - AAX_IParameter, [1069](#)
- SetBaseOffset
 - AAX_IDma, [1013](#)
- SetBurstLength
 - AAX_IDma, [1009](#)
- SetChunk
 - AAX_CEffectParameters, [495](#)
 - AAX_IACFEffEffectParameters, [828](#)
- SetCoefficients
 - AAX_Map, [1135](#)
- SetControlHighlightInfo
 - AAX_CEffectGUI, [466](#)
 - AAX_IACFEffEffectGUI, [805](#)
- SetCustomData
 - AAX_CEffectParameters, [500](#)
 - AAX_IACFEffEffectParameters, [830](#)
- SetCycleCount
 - AAX_IACFController, [771](#)
 - AAX_IController, [976](#)
 - AAX_VController, [1198](#)
- SetDefaultValue
 - AAX_CParameter< T >, [606](#)
- SetDirty
 - AAX_CPacket, [567](#)
 - AAX_CPacketDispatcher, [571](#)
- SetDisplayDelegate
 - AAX_CEffectParameters, [503](#)
 - AAX_CParameter< T >, [592](#)
 - AAX_CStatelessParameter, [684](#)
 - AAX_IParameter, [1084](#)
- SetDmaState
 - AAX_IDma, [1008](#)
- SetDone
 - AAX_IACFTask, [908](#)
 - AAX_ITask, [1115](#)
 - AAX_VTask, [1256](#)
- SetDst
 - AAX_IDma, [1009](#)
- SetFifoBuffer
 - AAX_IDma, [1011](#)
- SetFifoSize
 - AAX_IDma, [1014](#)
- SetLinearBuffer
 - AAX_IDma, [1012](#)
- SetLocation
 - AAX_CHostProcessor, [518](#)
 - AAX_IACFHostProcessor, [853](#)
- SetManufacturerName
 - AAX_IACFCollection, [748](#)
 - AAX_ICollection, [948](#)
 - AAX_VCollection, [1171](#)
- SetName
 - AAX_CParameter< T >, [585](#)
 - AAX_CStatelessParameter, [663](#)
 - AAX_IParameter, [1067](#)
- SetNormalizedDefaultValue
 - AAX_CParameter< T >, [587](#)
 - AAX_CStatelessParameter, [668](#)
 - AAX_IParameter, [1070](#)
- SetNormalizedValue
 - AAX_CParameter< T >, [588](#)
 - AAX_CStatelessParameter, [668](#)
 - AAX_IParameter, [1070](#)
- SetNumberOfSteps
 - AAX_CParameter< T >, [588](#)
 - AAX_CStatelessParameter, [669](#)
 - AAX_IParameter, [1071](#)
- SetNumBursts
 - AAX_IDma, [1010](#)
- SetNumOffsets
 - AAX_IDma, [1013](#)
- SetOffsetTable
 - AAX_IDma, [1012](#)

- SetOrientation
 - AAX_CParameter< T >, 591
 - AAX_CStatelessParameter, 683
 - AAX_IPParameter, 1083
- SetPackageVersion
 - AAX_IACFCollection, 749
 - AAX_ICollection, 949
 - AAX_VCollection, 1173
- SetParameterDefaultNormalizedValue
 - AAX_CEffectParameters, 482
 - AAX_IACFEffEffectParameters, 816
- SetParameterNameVariation
 - AAX_IACFPageTable_V2, 884
 - AAX_IPageTable, 1062
 - AAX_VPageTable, 1239
- SetParameterNormalizedRelative
 - AAX_CEffectParameters, 488
 - AAX_IACFEffEffectParameters, 821
- SetParameterNormalizedValue
 - AAX_CEffectParameters, 487
 - AAX_IACFEffEffectParameters, 821
- SetParameters
 - AAX_FastInterpolatedTableLookup< TFLOAT, DFLOAT >, 741
- SetPrimary
 - AAX_CStringAbbreviations, 707
- SetProgress
 - AAX_IACFTask, 907
 - AAX_ITask, 1114
 - AAX_VTask, 1255
- SetProperties
 - AAX_IACFCollection, 749
 - AAX_IACFEffEffectDescriptor, 791
 - AAX_ICollection, 949
 - AAX_IEffectDescriptor, 1018
 - AAX_VCollection, 1173
 - AAX_VEffectDescriptor, 1217
- SetSessionDocument
 - AAX_CSessionDocumentClient, 650
 - AAX_IACFSessionDocumentClient, 904
- SetSignalLatency
 - AAX_IACFController, 771
 - AAX_IController, 975
 - AAX_VController, 1197
- SetSrc
 - AAX_IDma, 1008
- SetStepValue
 - AAX_CParameter< T >, 590
 - AAX_CStatelessParameter, 670
 - AAX_IPParameter, 1072
- SetTaperDelegate
 - AAX_CEffectParameters, 503
 - AAX_CParameter< T >, 591
 - AAX_CStatelessParameter, 684
 - AAX_IPParameter, 1084
- SetToDefaultValue
 - AAX_CParameter< T >, 587
 - AAX_CStatelessParameter, 669
- AAX_IPParameter, 1071
- SetTransferSize
 - AAX_IDma, 1011
- SetType
 - AAX_CParameter< T >, 590
 - AAX_CStatelessParameter, 682
 - AAX_IPParameter, 1083
- SetValue
 - AAX_CParameter< T >, 605
- SetValueFromString
 - AAX_CParameter< T >, 599
 - AAX_CStatelessParameter, 678
 - AAX_IPParameter, 1078
- SetValueWithBool
 - AAX_CParameter< T >, 603, 607
 - AAX_CStatelessParameter, 680
 - AAX_IPParameter, 1081
- SetValueWithDouble
 - AAX_CParameter< T >, 604, 608
 - AAX_CStatelessParameter, 682
 - AAX_IPParameter, 1082
- SetValueWithFloat
 - AAX_CParameter< T >, 604, 608
 - AAX_CStatelessParameter, 681
 - AAX_IPParameter, 1082
- SetValueWithInt32
 - AAX_CParameter< T >, 603, 608
 - AAX_CStatelessParameter, 681
 - AAX_IPParameter, 1081
- SetValueWithString
 - AAX_CParameter< T >, 605, 609
 - AAX_CStatelessParameter, 682
 - AAX_IPParameter, 1082
- SetViewContainer
 - AAX_CEffectGUI, 463
 - AAX_IACFEffEffectGUI, 803
- SetViewSize
 - AAX_IACFViewContainer, 932
 - AAX_IViewContainer, 1130
 - AAX_VViewContainer, 1269
- SHORT_STRING_IDENTIFIER
 - AAX_ChunkDataParserDefs, 409
- SHORT_TYPE
 - AAX_ChunkDataParserDefs, 409
- SHORT_TYPE_INCR
 - AAX_ChunkDataParserDefs, 409
- SHORT_TYPE_SIZE
 - AAX_ChunkDataParserDefs, 409
- ShortenedName
 - AAX_CParameter< T >, 586
 - AAX_CStatelessParameter, 665
 - AAX_IPParameter, 1068
- Sidechain Inputs, 92
- Sign
 - AAX, 396
- SinCosMix
 - AAX, 396
- Size

- AAX_CArrayDataBuffer< D >, 419
- AAX_CArrayDataBufferOfType< T, D >, 423
- AAX_CStringDataBuffer, 713
- AAX_CStringDataBufferOfType< T >, 717
- AAX_IACFDataBuffer, 784
- AAX_IDataBufferWrapper, 989
- AAX_ISessionDocument::TempoMap, 1289
- AAX_VDataBufferWrapper, 1209
- AAX_VSessionDocument::VTempoMap, 1291
- SmartRound
 - AAX_CRangeTaperDelegate< T, RealPrecision >, 647
- StackTrace
 - AAX_CHostServices, 527
 - AAX_IACFHostServices_V2, 869
 - AAX_IHostServices, 1043
 - AAX_VHostServices, 1227
- StaticDescribe
 - AAX_CMonolithicParameters, 555
- StaticRenderAudio
 - AAX_CMonolithicParameters, 557
- StdString
 - AAX_CString, 695, 696
- String2Binary
 - AAX, 384
- STRING_IDENTIFIER_SIZE
 - AAX_ChunkDataParserDefs, 410
- STRING_STRING_IDENTIFIER
 - AAX_ChunkDataParserDefs, 409
- STRING_TYPE
 - AAX_ChunkDataParserDefs, 409
- StringToValue
 - AAX_CBinaryDisplayDelegate< T >, 434
 - AAX_CDecibelDisplayDelegateDecorator< T >, 451
 - AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >, 564
 - AAX_CPercentDisplayDelegateDecorator< T >, 635
 - AAX_CStateDisplayDelegate< T >, 657
 - AAX_CStringDisplayDelegate< T >, 722
 - AAX_CUnitDisplayDelegateDecorator< T >, 733
 - AAX_CUnitPrefixDisplayDelegateDecorator< T >, 739
 - AAX_IDisplayDelegate< T >, 995
 - AAX_IDisplayDelegateDecorator< T >, 1002
- SubString
 - AAX_CString, 701
- Supplemental Information, 285
- Supported
 - AAX_VDescriptionHost, 1212
- SupportLevel
 - AAX_IACFFeatureInfo, 849
 - AAX_IFeatureInfo, 1034
 - AAX_VFeatureInfo, 1220
- Taper delegates, 108
- TaperDelegate
 - AAX_CParameter< T >, 606
- Task agent interface, 103
 - AAX_TaskCompletionStatus, 104
 - Canceled, 104
 - Done, 104
 - Error, 104
 - None, 104
- template_size
 - AAX_CAtomicQueue< T, S >, 429
- template_type
 - AAX_CAtomicQueue< T, S >, 426
 - AAX_IPointerQueue< T >, 1090
- The Avid Component Framework (ACF), 144
- ThirtyTwoBitDSPCoefToDouble
 - AAX_CommonConversions.h, 1379
- TI_VERSION
 - AAX.h, 1306
- TimerWakeup
 - AAX_CEffectDirectData, 456
 - AAX_CEffectGUI, 464
 - AAX_CEffectParameters, 496
 - AAX_CMonolithicParameters, 555
 - AAX_IACFEffEffectDirectData, 796
 - AAX_IACFEffEffectGUI, 804
 - AAX_IACFEffEffectParameters, 829
- TimerWakeup_PrivateDataAccess
 - AAX_CEffectDirectData, 457
- ToDouble
 - AAX_CString, 701
- ToHexadecimal
 - AAX::internal, 407
- ToInteger
 - AAX_CString, 701
- Token protocol, 123
- top
 - AAX_Rect, 1139
- ToString
 - AAX_IMIDIMessageInfoDelegate, 1045
 - AAX_TransportStateInfo_V1, 1162
- ToString_AppendByteRange
 - AAX_IMIDIMessageInfoDelegate, 1047
- ToString_AppendCStr
 - AAX_IMIDIMessageInfoDelegate, 1046
- ToString_AppendNumber
 - AAX_IMIDIMessageInfoDelegate, 1046
- ToString_AppendValid
 - AAX_IMIDIMessageInfoDelegate, 1047
- Touch
 - AAX_CParameter< T >, 600
 - AAX_CStatelessParameter, 667
 - AAX_IParameter, 1069
- TouchParameter
 - AAX_CEffectParameters, 488
 - AAX_IACFEffEffectParameters, 822
- TParamValPair
 - AAX_CMonolithicParameters, 549
- Trace
 - AAX_CHostServices, 526
 - AAX_IACFHostServices, 867

- AAX_IHostServices, [1043](#)
- AAX_VHostServices, [1227](#)
- TranslateOutputBounds
 - AAX_CHostProcessor, [523](#)
- Transport
 - AAX_CEffectGUI, [468](#)
 - AAX_CEffectParameters, [502](#)
- Troubleshooting, [286](#)
- Try_Lock
 - AAX_CMutex, [560](#)
- Type
 - AAX_CArrayDataBuffer< D >, [419](#)
 - AAX_CArrayDataBufferOfType< T, D >, [423](#)
 - AAX_CParameter< T >, [580](#)
 - AAX_CStringDataBuffer, [713](#)
 - AAX_CStringDataBufferOfType< T >, [717](#)
 - AAX_IACFDataBuffer, [784](#)
 - AAX_IDataBufferWrapper, [989](#)
 - AAX_VDataBufferWrapper, [1208](#)
- Uninitialize
 - AAX_CEffectDirectData, [455](#)
 - AAX_CEffectGUI, [462](#)
 - AAX_CEffectParameters, [479](#)
 - AAX_CHostProcessor, [516](#)
 - AAX_CSessionDocumentClient, [650](#)
 - AAX_CTaskAgent, [726](#)
 - AAX_IACFEffectDirectData, [796](#)
 - AAX_IACFEffectGUI, [802](#)
 - AAX_IACFEffectParameters, [812](#)
 - AAX_IACFHostProcessor, [852](#)
 - AAX_IACFSessionDocumentClient, [904](#)
 - AAX_IACFTaskAgent, [911](#)
- Unlock
 - AAX_CMutex, [559](#)
- UnregisterParameter
 - AAX_IACFAutomationDelegate, [744](#)
 - AAX_IAutomationDelegate, [942](#)
 - AAX_VAutomationDelegate, [1165](#)
- UpdateAllParameters
 - AAX_CEffectGUI, [467](#)
- UpdateControlMIDINodes
 - AAX_CEffectParameters, [501](#)
 - AAX_IACFEffectParameters_V2, [836](#)
- UpdateMIDINodes
 - AAX_CEffectParameters, [500](#)
 - AAX_IACFEffectParameters_V2, [835](#)
- UpdateNormalizedValue
 - AAX_CParameter< T >, [605](#)
 - AAX_CStatelessParameter, [684](#)
 - AAX_IPParameter, [1085](#)
- UpdatePageTable
 - AAX_CEffectParameters, [499](#), [504](#)
 - AAX_IACFEffectParameters_V4, [847](#)
- UpdateParameterNormalizedRelative
 - AAX_CEffectParameters, [490](#)
 - AAX_IACFEffectParameters, [824](#)
- UpdateParameterNormalizedValue
 - AAX_CEffectParameters, [490](#)
- AAX_CMonolithicParameters, [552](#)
- AAX_IACFEffectParameters, [824](#)
- UpdateParameterTouch
 - AAX_CEffectParameters, [489](#)
 - AAX_IACFEffectParameters, [823](#)
- Valid
 - AAX_ISessionDocument, [1101](#)
 - AAX_VSessionDocument, [1251](#)
- value_type
 - AAX_CAtomicQueue< T, S >, [427](#)
 - AAX_IPointerQueue< T >, [1090](#)
- ValueToString
 - AAX_CBinaryDisplayDelegate< T >, [433](#)
 - AAX_CDecibelDisplayDelegateDecorator< T >, [449](#), [450](#)
 - AAX_CNumberDisplayDelegate< T, Precision, SpaceAfter >, [562](#), [563](#)
 - AAX_CPercentDisplayDelegateDecorator< T >, [633](#), [634](#)
 - AAX_CStateDisplayDelegate< T >, [656](#), [657](#)
 - AAX_CStringDisplayDelegate< T >, [721](#)
 - AAX_CUnitDisplayDelegateDecorator< T >, [731](#), [732](#)
 - AAX_CUnitPrefixDisplayDelegateDecorator< T >, [738](#)
 - AAX_IDisplayDelegate< T >, [994](#), [995](#)
 - AAX_IDisplayDelegateDecorator< T >, [1000](#), [1001](#)
- VENUE Guide, [355](#)
- VERSION_ID_1
 - AAX_ChunkDataParserDefs, [410](#)
- vert
 - AAX_Point, [1138](#)
- VTempoMap
 - AAX_VSessionDocument::VTempoMap, [1291](#)
- What
 - AAX::Exception::Any, [1275](#)
- width
 - AAX_Rect, [1139](#)
- WordAlign
 - AAX_CChunkDataParser, [445](#)
- WritePortDirect
 - AAX_IACFPrivateDataAccess, [892](#)
 - AAX_IPrivateDataAccess, [1094](#)
 - AAX_VPrivateDataAccess, [1243](#)
- ZeroMemoryDW
 - AAX, [392](#)
- ZeroMemorySW
 - AAX, [392](#)