After Effects CC 2015

SDK Guide

#### Release 1

July 21, 2015

Copyright © 1992–2015 Adobe Systems Incorporated. All rights reserved.

The information in this document is furnished for informational use only, is subject to

change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in this document. The software described in this

document is furnished under license and may only be used or copied in accordance with the terms of such license.

Adobe, Adobe After Effects, Adobe Premiere Pro, Adobe Photoshop, Adobe Illustrator,

Adobe Type Manager, ATM and PostScript are trademarks of Adobe Systems Incorporated that may be registered in certain jurisdictions. Macintosh and Apple are registered

trademarks, and Mac OS are trademarks of Apple Computer, Inc. Microsoft, Excel, and Windows are registered trademarks of Microsoft Corporation. All other products or name brands are trademarks of their respective holders.

The material in this document is supplied by the Adobe Digital Video API Engineering team.

**TABLE 1: VERSION HISTORY**

|  |  |  |
| --- | --- | --- |
| Version History | | |
| January 1993 | Russell Belfer | Version 1.0 – Initial SDK release. |
| January 1994 | Dan Wilk | Version 2.0 – Updates. |
| August 1994 | Dave Herbstman Dan Wilk | Version 2.0.1 – Added support for PowerPC. |
| 5 March 1996 | Brian Andrews | Version 3.0 – Preliminary release for the After Effects developer kitchen. |
| 21 June 1996 | Brian Andrews | Version 3.1 – Final 3.x release. |
| 13 Nov. 1996 | Brian Andrews | Version 3.1 SDK Release 2 – Minor updates. |
| 17 April 1997 | Brian Andrews | Version 3.1 SDK Release 3 – First public release (really a pre- release) of the SDK for Windows development. |
| 1 May 1998 | Bruce Bullis | Version 3.1 SDK Release 6 - Editorial changes only |
| 1 January 1999 | Bruce Bullis | Version 4.0 SDK Release 1 - Added information on new global flags, custom data types, utilization of PICA suites, CustomUI messaging and parameter supervision, new callbacks. many editorial changes. |
| 9 September 1999 | Bruce Bullis | Revised for 4.1; added General plug-ins and AEGP information. Added information on new selectors, resize handle. |

|  |  |  |
| --- | --- | --- |
| Version History | | |
| 2 February 2001 | Bruce Bullis | 5.0 release. Entire document edited and reformatted. Sections on 16 bit-per-channel color and parameter supervision, as well as the entire AEGP chapter, have all substantially expanded. |
| 1 December 2001 | Bruce Bullis | 5.5 release. Added information on new outflags, PiPL changes, and additions and changes to the AEGP API. Numerous clarifications and edits. |
| 4 March 2002 | Bruce Bullis | Updated Mac OS X details, expanded AEIO and AEGP documentation. |
| 20 July 2003 | Bruce Bullis | Major overhauls for After Effects 6.0. Added documentation for all new (and some old) suites, and many supporting details for effects. |
| 4 April 2004 | Bruce Bullis | Updated for 6.5. Expanded and corrected all documentation. Added documentation of all new AEGP functions. |
| 1 December 2005 | Bruce Bullis | Updated for 7.0. Added SmartFX documentation. Noted current suite version numbers throughout. Numerous editorial changes. Documented many new AEGP suite functions. |
| 4 April 2006 | Bruce Bullis | Updated to reference new development system requirements and XCode-specific issues. Some editing. |
| 1 July 2007 | Bruce Bullis | CS3 (8.0) release. |
| 4 May 2009 | Zac Lam | CS4 (9.0) release. Complete reorganization of first three chapters. Fleshed out documentation on Premiere Pro. |
| 28 April 2010 | Zac Lam | CS5 (10.0) release. 64-bit porting info. Drawbot. |
| 2 May 2011 | Zac Lam | CS5.5 (10.5) release. |
| 26 April 2012 | Zac Lam | CS6 (11.0) release. Big reorganization of the AEGP and Artisan chapters. Many additions throughout. |
| 15 July 2013 | Zac Lam | CC (12.0) release. API version changes, misc small clarifications, more details in AEIO chapter. |
| 12 June 2014 | Zac Lam | CC 2014 (13.0) release. Corrections for accuracy. Removed old version references. |
| 21 July 2015 | Zac Lam | CC 2015 (13.5) release. |

### ABOUT THIS DOCUMENT

This document has changed much over the years. Part encyclopedia, part how-to guide, with multiple sedimentary layers of accreted information from more than two decades of API

development and refinement. Yes, there does need to be one source of information about

every last niggling detail of the After Effects APIs. However, since no human in their right mind would ever want to *read* such a document, we’ve tried to keep it involving and

interesting. As opportunity allows, we’ll try to include more diagrams, illustrations, and

purdy pickshurs explaining API intricacies. As always, your input is valued and appreciated.

**ORGANIZATION**

The [Introduction](#_bookmark9) provides an overview of the integration possibilities with After Effects. It explains what plug-ins are, and how they work with After Effects. It describes the sample

projects, and how to modify them. It explains where to install plug-ins, and what resources they use.

The basics of effect plug-ins are discussed in [chapter 2](#_bookmark68). This overview provides information on the function parameters passed to and from an effect plug-in’s entry point. It describes

capability flags, effect parameters, and image buffers.

[Chapter 3](#_bookmark254) dives into the details of developing a complete effect plug-in using the many provided callback functions. It also provides many testing ideas to ensure the plug-in is stabile.

[SmartFX](#_bookmark402) is the extension to the effect plug-in API to support 32-bit floating point images, and is covered in chapter 4.

Chapter 5 covers [events](#_bookmark425) sent to effect plug-ins, how to incorporate custom user interface

elements, parameter supervision, and the reliance of custom data parameter types on Custom UI messaging.

[Audio](#_bookmark514) effects are covered in Chapter 6.

Chapter 7 details the After Effects General Plug-in ([AEGP](#_bookmark528)) API. Provided callback

functions, hooking into internal messaging, manipulating the current contents of open projects and handling menu commands are all covered at length.

[Artisans](#_bookmark731), specialized plug-in 3D renderer AEGPs, are covered in chapter 8. Under no circumstances should anyone write an artisan, ever.

Chapter 9 documents [AEIOs](#_bookmark768), specialized AEGPs which handle file input and output.

Chapter 10 discusses issues related to compatibility with Premiere Pro and other applications that support a subset of After Effects plug-ins.

**DOCUMENTATION CONVENTIONS**

Functions, structure names and general C/C++ code are in Courier; MyStruct and

MyFunction();

Text in blue is hyperlinked.

Command selectors are italicized; *PF\_Cmd\_RENDER*.

**A NOTE ABOUT CODING STYLE**

Because we use the public APIs for our own plug-ins, our coding guidelines are apparent

throughout the SDK. Here’s a description of the pseudo-neo-post-Hungarian notation we use. Of course, you’re welcome to code however you like. If you feel strongly that we should change our internal coding standards, please post your requests at comp.sys.programmer.better.things.to.do.with.your.time, and we’ll

carefully consider them before not making any changes.

**TABLE 2: CODING CONVENTIONS**

|  |  |  |
| --- | --- | --- |
| Type | Suffix | Example |
| Handle | **H** | fooH |
| pointer (to) | **P** | fooP |
| Boolean | **B** | visibleB |
| Float | **F** | degreesF |
| Long | **L** | offsetL |
| unsigned long | **Lu** | countLu |
| short | **S** | indexS |
| char | **C** | digitC |
| unsigned char | **Cu** | redCu |
| function pointer | **\_func** | sample\_func |
| time value | **T** | durationT |
| char \* (NULL-terminated C string) | **Z** | nameZ |
| rectangle | **R** | boundsR |

**TABLE 2: CODING CONVENTIONS**

|  |  |  |
| --- | --- | --- |
| Type | Suffix | Example |
| fixed rectangle | **FiR** | boundsFiR |
| float rectangle | **FR** | boundsFR |
| ratio | **Rt** | scale\_factorRt |
| void \* | **PV** | refconPV |
| optional parameter (must be passed, can be NULL) | **0** | extra\_flags0 |

[Table 1: Version History 2](#_bookmark0)

[About this document 4](#_bookmark1)

[Organization 4](#_bookmark2)

[Documentation Conventions 5](#_bookmark3)

[A note about coding style 5](#_bookmark4)

[Table 2: Coding conventions 5](#_bookmark5)

[CHAPTER 1: INTRODUCTION 22](#_bookmark6)

[What can i do with this SDK? 22](#_bookmark7)

[What plug-ins can i build with this SDK? 22](#_bookmark8)

[Where do plug-ins appear in After Effects? 23](#_bookmark10)

[How does After Effects interact with plug-ins? 24](#_bookmark11)

[SDK contents 24](#_bookmark12)

[Other integration possibilities 25](#_bookmark13)

[Scripting 25](#_bookmark14)

[HTML5 Panels 25](#_bookmark15)

[aerender 26](#_bookmark16)

[Premiere Pro Importers 26](#_bookmark17)

[Mercury Transmit 26](#_bookmark18)

[SDK Audience 26](#_bookmark19)

[Development Requirements 27](#_bookmark20)

[What’s New? 27](#_bookmark21)

[What’s New in CC 2015.0.1 (13.5.1)? 27](#_bookmark23)

[What’s New in CC 2015 (13.5)? 28](#_bookmark24)

[What’s New in CC 2014 (13.0)? 33](#_bookmark25)

[What’s New in CC (12.0)? 34](#_bookmark26)

[What’s New in CS6.0.1 (11.0.1)? 34](#_bookmark27)

[What’s New in CS6 (11.0)? 35](#_bookmark28)

[...and what was new before CS6? 36](#_bookmark29)

[How to start creating plug-ins 36](#_bookmark30)

[Play! 36](#_bookmark31)

[Plan! 36](#_bookmark32)

[Hack! 37](#_bookmark33)

[Steal! 37](#_bookmark34)

[Test! 37](#_bookmark35)

[Blame! 37](#_bookmark36)

[Developers matter 37](#_bookmark37)

[Sample projects 38](#_bookmark38)

[Table 3: Sample Project Descriptions 38](#_bookmark39)

[Building the sample projects 40](#_bookmark40)

[Debugging plug-ins 41](#_bookmark41)

[Deleting Preferences 42](#_bookmark42)

[Compatibility across multiple versions? 42](#_bookmark44)

[Table 4: API Versions 42](#_bookmark46)

[Third-party plug-in hosts? 43](#_bookmark47)

[PiPL resources 43](#_bookmark48)

[Entry Point 44](#_bookmark50)

[PiPL resources and Microsoft Visual Studio 44](#_bookmark51)

[Multiple PiPLs 44](#_bookmark52)

[Super Secret PiPL bit 45](#_bookmark53)

[Why do I need to know all this? 45](#_bookmark54)

[Exceptions 45](#_bookmark55)

[Where installers should put plug-ins 45](#_bookmark56)

[Do I have to install the plug-ins to the common folder? 46](#_bookmark58)

[Localization 47](#_bookmark59)

[Next steps 47](#_bookmark60)

[CHAPTER 2: EFFECT BASICS 48](#_bookmark61)

[Entry Point 48](#_bookmark62)

[Table 5: Entry Point Function Parameters 48](#_bookmark65)

[Command Selectors 49](#_bookmark70)

[Calling sequence 49](#_bookmark71)

[Table 6: Command Selectors 50](#_bookmark72)

[What’s the difference? 55](#_bookmark109)

[PF\_InData 56](#_bookmark110)

[Table 7: PF\_InData 56](#_bookmark112)

[extent\_hint usage 59](#_bookmark124)

[Now with 20% more pixels! 60](#_bookmark125)

[Point controls and buffer expansion 60](#_bookmark126)

[PF\_OutData 61](#_bookmark127)

[Table 8: PF\_OutData 61](#_bookmark129)

[PF\_OutFlags 62](#_bookmark142)

[Table 9: PF\_OutFlags 62](#_bookmark144)

[PF\_OutFlags2 66](#_bookmark166)

[Table 10: PF\_OutFlags2 66](#_bookmark168)

[Parameters 69](#_bookmark184)

[Table 11: Parameter Types 70](#_bookmark188)

[Slider range issues? 73](#_bookmark204)

[Point parameter origin 73](#_bookmark205)

[PF\_ParamDef 73](#_bookmark206)

[Param zero 73](#_bookmark208)

[Table 12: PF\_ParamDef 74](#_bookmark210)

[Parameter UI Flags 75](#_bookmark211)

[Table 13: Parameter UI Flags 75](#_bookmark213)

[Parameter Flags 77](#_bookmark216)

[Table 14: Parameter Flags 77](#_bookmark217)

[PF\_ValueDisplayFlags 78](#_bookmark225)

[PF\_EffectWorld / PF\_LayerDef 79](#_bookmark226)

[Table 15: PF\_EffectWorld structure 79](#_bookmark229)

[Rowbytes in PF\_EffectWorlds 80](#_bookmark239)

[Byte alignment 80](#_bookmark240)

[Deeper color 80](#_bookmark241)

[Accessor macros for opaque (data type) pixels 81](#_bookmark242)

[Table 16: PF\_PixelPtr accessor macros 81](#_bookmark243)

[Errors 82](#_bookmark246)

[Table 17: Error Codes 82](#_bookmark247)

[Error reporting policy 82](#_bookmark249)

[Dig in! 83](#_bookmark250)

[CHAPTER 3: EFFECT DETAILS 84](#_bookmark251)

[Free code == GOOD 84](#_bookmark252)

[Accessing the After Effects function suites 84](#_bookmark253)

[Suite Versions 85](#_bookmark255)

[Threading 85](#_bookmark256)

[Memory allocation 85](#_bookmark257)

[Table 18: PF\_HandleSuite1 86](#_bookmark259)

[Image buffer management functions 87](#_bookmark260)

[Table 19: PF\_WorldSuite2 87](#_bookmark261)

[Iteration suites 87](#_bookmark262)

[Table 20: PF\_Iterate8Suite1, PF\_Iterate16Suite1, PF\_IterateFloatSuite1 88](#_bookmark263)

[Graphics utility suites 90](#_bookmark264)

[Transform Worlds 91](#_bookmark265)

[Table 21: PF\_WorldTransformSuite1 91](#_bookmark266)

[Kernel Flags 93](#_bookmark269)

[Table 22: Kernel Flags 94](#_bookmark270)

[Fill ‘em up! 95](#_bookmark271)

[Table 23: PF\_FillMatteSuite2 95](#_bookmark272)

[Sampling images 96](#_bookmark273)

[Table 24: PF\_SamplingSuite Functions (multiple suites) 96](#_bookmark274)

[Table 25: PF\_BatchSamplingSuite1 97](#_bookmark275)

[Do the math for me 98](#_bookmark276)

[Table 26: PF\_ANSICallbacksSuite1 98](#_bookmark278)

[Interaction callback functions 100](#_bookmark279)

Table 27: [Interaction Callbacks 100](#_bookmark282)

[Parameter checkout vs. param zero 103](#_bookmark297)

[Parameter checkout behavior 103](#_bookmark298)

[Parameter checkout and re-entrancy 104](#_bookmark299)

[Progress during iteration 104](#_bookmark300)

[Pixel aspect ratio 105](#_bookmark301)

[Don’t assume pixels are square, or 1-to-1 105](#_bookmark302)

[Suggested approach 106](#_bookmark303)

[Applying user input in pixels 107](#_bookmark304)

[Test test test! 107](#_bookmark305)

[Parameters and floating point values 107](#_bookmark306)

[Table 28: PF\_ColorParamSuite1 108](#_bookmark308)

[Table 29: PF\_PointParamSuite1 108](#_bookmark310)

[Table 30: PF\_AngleParamSuite1 108](#_bookmark312)

[Parameter supervision 108](#_bookmark313)

[Updating parameter UI 109](#_bookmark315)

[Updating parameter values 109](#_bookmark316)

[Parameter Utility Suite 110](#_bookmark317)

[Table 31: PF\_ParamUtilsSuite3 110](#_bookmark319)

[Global, sequence, and frame data 114](#_bookmark328)

[Persistence 114](#_bookmark329)

[Validating Sequence Data 114](#_bookmark330)

[Flattened and unflattened sequence data 115](#_bookmark332)

[Resizing sequence data 115](#_bookmark333)

[Arbitrary data parameters 116](#_bookmark334)

[Table 32: Arbitrary data selectors 116](#_bookmark336)

[Implementing arbitrary data 117](#_bookmark341)

[Arbitrary data? Re-entrancy. 118](#_bookmark342)

[When NOT to access arbitrary parameters 118](#_bookmark343)

[Changes during dialogs 118](#_bookmark344)

[Useful utility functions 119](#_bookmark345)

[PF\_EffectUISuite 119](#_bookmark346)

[Table 33: PF\_EffectUISuite 119](#_bookmark347)

[PF\_AppSuite 119](#_bookmark348)

[Table 34: PF\_AppSuite5 120](#_bookmark350)

[Advanced AppSuite: you can](#_bookmark353) *do* that?! 122

[Table 35: AE\_AdvAppSuite2 122](#_bookmark354)

[Formatting time 124](#_bookmark355)

[Table 36: PF\_AdvTimeSuite2 124](#_bookmark356)

[Affecting the timeline 125](#_bookmark357)

[Table 37: PF\_AdvItemSuite1 126](#_bookmark358)

[Accessing auxiliary channel data 127](#_bookmark359)

[Table 38: PF\_ChannelSuite1 127](#_bookmark360)

[Motion blur 129](#_bookmark361)

[Working with paths 129](#_bookmark363)

[Accessing path data 129](#_bookmark364)

[Manipulating path data 129](#_bookmark365)

[Vertices 130](#_bookmark366)

[Table 39: PF\_PathVertex 130](#_bookmark367)

[PF\_PathDataSuite 130](#_bookmark368)

[Table 40: PF\_PathDataSuite1 130](#_bookmark370)

[PF\_PathQuerySuite 133](#_bookmark371)

[Table 41: PF\_PathQuerySuite 133](#_bookmark373)

[Accessing camera and light information 134](#_bookmark374)

[Color space conversion 135](#_bookmark375)

[Table 42: Pixel Types for different color spaces 135](#_bookmark376)

[Table 43: color space conversion callbacks 135](#_bookmark381)

[Changing parameter orders, the nice way 136](#_bookmark383)

[Change defaults? Change IDs 137](#_bookmark384)

[Tips and tricks 137](#_bookmark385)

[Best practices 137](#_bookmark386)

[Responsiveness 137](#_bookmark387)

[Make your effect easy to find 138](#_bookmark388)

[Sampling pixels at (x,y) 138](#_bookmark389)

[Where’s the center of a pixel? 138](#_bookmark390)

[Clean slate 139](#_bookmark391)

[Caching behavior 139](#_bookmark392)

[Global Performance Cache Consideratons 139](#_bookmark393)

[Some thoughts on time from a long-time developer 140](#_bookmark395)

[Rate x Time == PAIN 142](#_bookmark396)

[Testing 142](#_bookmark398)

[CHAPTER 4: SMARTFX 143](#_bookmark399)

[The way things were 143](#_bookmark400)

[The way things are now 143](#_bookmark401)

[Content Bounds 144](#_bookmark403)

[How to Smartify 144](#_bookmark404)

[PF\_Cmd\_SMART\_PRE\_RENDER 144](#_bookmark405)

[Table 44: PF\_PreRenderExtra 145](#_bookmark406)

[preserve\_rgb\_of\_zero\_alpha 148](#_bookmark411)

[rectangles 148](#_bookmark412)

[Table 45: PF\_PreRenderOutput 148](#_bookmark410)

[The “size” of a layer 149](#_bookmark413)

[Flag on the play 150](#_bookmark414)

[PF\_Cmd\_SMART\_RENDER 150](#_bookmark415)

[Table 46: PF\_SmartRenderExtra 150](#_bookmark416)

[When to access layer parameters 151](#_bookmark417)

[Wait, Gimme That Layer Back! 152](#_bookmark418)

[CHAPTER 5: EFFECT UI & EVENTS 153](#_bookmark419)

[Table 47: Events 153](#_bookmark421)

[PF\_EventExtra 154](#_bookmark434)

Table 48: [PF\_EventExtra 154](#_bookmark436)

[PF\_Context 156](#_bookmark444)

[Table 49: PF\_Context 156](#_bookmark445)

[Table 50: PF\_EffectWindowInfo 156](#_bookmark446)

[PF\_EventUnion 157](#_bookmark454)

[Click 157](#_bookmark455)

[Table 51: PF\_DoClickEventInfo 157](#_bookmark456)

[Draw 157](#_bookmark464)

[Table 52: PF\_DrawEventInfo 157](#_bookmark467)

[Keydown 158](#_bookmark470)

[Table 53: PF\_KeyDownEvent 158](#_bookmark472)

[AdjustCursor 159](#_bookmark473)

[Table 54: PF\_AdjustCursorEventInfo 159](#_bookmark474)

[Arbitrary Parameters Event 159](#_bookmark475)

[Table 55: PF\_ArbParamsExtra 159](#_bookmark476)

[Custom UI and Drawbot 160](#_bookmark477)

[Make Your Custom UI Look Not So “Custom” 161](#_bookmark478)

[Redrawing 161](#_bookmark479)

[HiDPI and Retina Display Support 161](#_bookmark481)

[PF\_EffectCustomUISuite 161](#_bookmark482)

[Table 56: PF\_EffectCustomUISuite1 161](#_bookmark483)

[DRAWBOT\_DrawbotSuite 162](#_bookmark485)

[Table 57: DRAWBOT\_DrawbotSuite1 162](#_bookmark486)

[DRAWBOT\_SupplierSuite 162](#_bookmark487)

[Table 58: DRAWBOT\_SupplierSuite1 162](#_bookmark488)

[DRAWBOT\_SurfaceSuite 164](#_bookmark490)

[Table 59: DRAWBOT\_SurfaceSuite1 164](#_bookmark491)

[DRAWBOT\_PathSuite 167](#_bookmark493)

[Table 60: DRAWBOT\_PathSuite1 167](#_bookmark494)

[PF\_EffectCustomUIOverlayThemeSuite 168](#_bookmark495)

Table 61: [PF\_EffectCustomUIOverlayThemeSuite1 168](#_bookmark497)

[UI callbacks 169](#_bookmark498)

[Table 62: UI Callbacks 170](#_bookmark499)

[Tips and tricks 172](#_bookmark503)

[UI Performance 172](#_bookmark504)

[No more black 173](#_bookmark505)

[How deep are my pixels? 173](#_bookmark506)

[Arbitrary data 173](#_bookmark507)

[Custom UI implementation for color sampling, using keyframes 173](#_bookmark508)

[CHAPTER 6: AUDIO 174](#_bookmark510)

[Global Outflags 174](#_bookmark511)

[Audio Data Structures 174](#_bookmark512)

[Table 63: Audio data structures 174](#_bookmark513)

[Audio-specific Float Slider Variables 175](#_bookmark515)

[Flags 175](#_bookmark517)

[Phase 175](#_bookmark518)

[Curve Tolerance 175](#_bookmark519)

[What’s zero, really? 175](#_bookmark520)

[Accessing Audio Data 176](#_bookmark521)

[Extending Audio Clips 176](#_bookmark522)

[Audio Considerations 176](#_bookmark524)

[CHAPTER 7: AEGPS 177](#_bookmark525)

[What’s New? 177](#_bookmark526)

[What’s New in CS6? 177](#_bookmark527)

[Overview 178](#_bookmark529)

[AEGP communication with After Effects 178](#_bookmark530)

[Different tasks, same API 178](#_bookmark531)

[Data Types 179](#_bookmark532)

Table 64: [AEGP API Data Types 179](#_bookmark533)

[Nasty, brutish, and short 181](#_bookmark534)

[Were you just going to](#_bookmark535) *leave* that data *lying around*? 181

[Table 65: Data types requiring disposal 181](#_bookmark536)

[Implementation 182](#_bookmark537)

[Entry Point 182](#_bookmark538)

[The Hook-Up 183](#_bookmark540)

[Specialization 183](#_bookmark541)

[Example: adding a menu item 183](#_bookmark542)

[Private Data 183](#_bookmark543)

[Threading 184](#_bookmark544)

[AEGP Suites 184](#_bookmark545)

[Table 66: AEGP Suites 184](#_bookmark546)

[Fail gracefully 186](#_bookmark547)

[Handling Handles 186](#_bookmark548)

[Table 67: AEGP\_MemorySuite1 187](#_bookmark550)

[Managing menu items 188](#_bookmark551)

[Table 68: AEGP\_CommandSuite1 188](#_bookmark552)

[Registering with After Effects 191](#_bookmark556)

[Table 69: AEGP\_RegisterSuite5 191](#_bookmark557)

[Manage Projects 193](#_bookmark562)

[Table 70: AEGP\_ProjSuite6 193](#_bookmark563)

[Table 71: AEGP\_TimeDisplay2 195](#_bookmark565)

[Control Items within projects 196](#_bookmark566)

[Table 72: AEGP\_ItemSuite8 196](#_bookmark568)

[Managing selections 201](#_bookmark570)

[Table 73: AEGP\_CollectionSuite2 202](#_bookmark571)

[Ownership of collection items 203](#_bookmark573)

[Manipulate Compositions 203](#_bookmark574)

[Table 74: AEGP\_CompSuite10 203](#_bookmark575)

[Work With Footage 210](#_bookmark581)

Table 75: [AEGP\_FootageSuite5 210](#_bookmark582)

[Table 76: AEGP\_FootageInterp structure 217](#_bookmark589)

[Manage Layers 218](#_bookmark590)

[Table 77: AEGP\_LayerSuite8 218](#_bookmark592)

[Layer creation notes 227](#_bookmark604)

[A note about layer offsets 228](#_bookmark605)

[Communication with a layer’s effects 228](#_bookmark606)

[Table 78: AEGP\_EffectSuite4 228](#_bookmark608)

[Exploiting effect UI behavior to look cool 232](#_bookmark613)

[StreamRefs and EffectRefs 232](#_bookmark614)

[Diving Into Streams! 233](#_bookmark615)

[Okay, what did I just get? 234](#_bookmark616)

[Layers 234](#_bookmark617)

[Masks 234](#_bookmark618)

[Effects 234](#_bookmark619)

[Stream Suite 235](#_bookmark620)

[Table 79: AEGP\_StreamSuite4 235](#_bookmark622)

[Dynamic Streams 242](#_bookmark632)

[Table 80: AEGP\_DynamicStreamSuite4 242](#_bookmark633)

[Working with keyframes 250](#_bookmark639)

[Table 81: AEGP\_KeyframeSuite3 250](#_bookmark641)

[Adding multiple keyframes 255](#_bookmark648)

[Marker Streams 255](#_bookmark649)

[Table 82: AEGP\_MarkerSuite2 255](#_bookmark650)

[Mask Management 258](#_bookmark651)

[Table 83: AEGP\_MaskSuite6 258](#_bookmark653)

[Mask Outlines 261](#_bookmark657)

[Table 84: AEGP\_MaskOutlineSuite3 261](#_bookmark659)

[Mask Feathering 263](#_bookmark660)

[Working with text layers 264](#_bookmark661)

[Table 85: AEGP\_TextDocumentSuite1 264](#_bookmark662)

[Working with text outlines 265](#_bookmark663)

[Table 86: AEGP\_TextLayerSuite1 265](#_bookmark664)

[Utility functions 266](#_bookmark665)

[Table 87: AEGP\_UtilitySuite6 266](#_bookmark666)

[Persistent Data Suite 270](#_bookmark672)

[Table 88: AEGP\_PersistentDataSuite4 271](#_bookmark674)

[Color Management 275](#_bookmark675)

[Table 89: AEGP\_ColorSettingsSuite2 275](#_bookmark676)

[Render Suites 277](#_bookmark680)

[Table 90: AEGP\_RenderOptionsSuite4 277](#_bookmark681)

[Table 91: AEGP\_LayerRenderOptionsSuite1 (New in 13.0) 281](#_bookmark682)

[Table 92: AEGP\_RenderSuite4 284](#_bookmark684)

[The AEGP\_World as we know it 288](#_bookmark689)

[Table 93: AEGP\_WorldSuite3 288](#_bookmark690)

[Track Mattes and Transform functions 291](#_bookmark693)

[Table 94: AEGP\_CompositeSuite2 291](#_bookmark694)

[Work With Audio 293](#_bookmark695)

[Table 95: AEGP\_SoundDataSuite1 293](#_bookmark696)

[Audio Settings 293](#_bookmark697)

[Render Queue Suite 294](#_bookmark698)

[Table 96: AEGP\_RenderQueueSuite1 294](#_bookmark699)

[Render Queue Item Suite 295](#_bookmark701)

[Table 97: AEGP\_RQItemSuite3 295](#_bookmark702)

[Render Queue Monitor Suite 297](#_bookmark703)

[Table 98: AEGP\_RenderQueueMonitorSuite1 297](#_bookmark705)

[Output Module Suite 302](#_bookmark706)

[Table 99: AEGP\_OutputModuleSuite4 302](#_bookmark707)

[Working with Effects 306](#_bookmark708)

[Table 100: AEGP\_PFInterfaceSuite1 307](#_bookmark710)

[AEGP\_GetEffectCameraMatrix notes 308](#_bookmark712)

[Do this many times 308](#_bookmark713)

Table 101: [AEGP\_IterateSuite1 308](#_bookmark714)

[File Import Manager suite 309](#_bookmark715)

[Table 102: AEGP\_FIMSuite3 309](#_bookmark716)

[Cheating: effect usage of AEGP suites 309](#_bookmark717)

[Depending on AEGP Queries 310](#_bookmark719)

[AEGP Details 310](#_bookmark720)

[Have a cookie 310](#_bookmark721)

[Modifying items in the render queue 310](#_bookmark722)

[Names and Solids 311](#_bookmark723)

[Reporting errors and problems 311](#_bookmark724)

[Transforms: what happens first? 311](#_bookmark725)

[Accessing pixels from effect layer parameters 311](#_bookmark726)

[CHAPTER 8: ARTISANS 312](#_bookmark727)

[Interactive Artisans 312](#_bookmark728)

[Artisan Data Types 312](#_bookmark729)

[Table 103: Data types used in the Artisan API 312](#_bookmark730)

[Horz? Vert? 313](#_bookmark732)

[Implementation and Design 313](#_bookmark733)

[3D compositing, not modeling 313](#_bookmark734)

[Registering an Artisan 313](#_bookmark735)

[Table 104: Artisan Entry Points 314](#_bookmark736)

[The World Is Your Canvas 316](#_bookmark739)

[Table 105: AEGP\_CanvasSuite8 317](#_bookmark741)

[Convert between different contexts 326](#_bookmark747)

[Table 106: AEGP\_ArtisanUtilSuite1 326](#_bookmark748)

[Smile! Cameras 327](#_bookmark749)

[Table 107: AEGP\_CameraSuite2 327](#_bookmark750)

[Notes regarding camera behavior 328](#_bookmark751)

[Orthographic camera matrix 328](#_bookmark752)

[Focus on Focal 328](#_bookmark753)

[Film size 328](#_bookmark754)

[Hit the lights! 329](#_bookmark755)

[Table 108: AEGP\_LightSuite2 329](#_bookmark756)

[Notes on light behavior 329](#_bookmark757)

[How should I draw that? 329](#_bookmark758)

[Transform Conventions 330](#_bookmark759)

[Query transform functions 330](#_bookmark760)

[Table 109: AEGP\_QueryXformSuite2 330](#_bookmark761)

[Interactive drawing functions 333](#_bookmark762)

[Table 110: PR\_InteractiveDrawProcs 333](#_bookmark763)

[Notes on Query Time functions 333](#_bookmark764)

[CHAPTER 9: AEIOS 334](#_bookmark765)

[AEIO, or AEGP? 334](#_bookmark766)

[AEIO for import, or MediaCore Importer? 334](#_bookmark767)

[How it Works 335](#_bookmark769)

[What would After Effects do? 335](#_bookmark770)

[Registering Your AEIO 335](#_bookmark771)

[InSpec, OutSpec 335](#_bookmark773)

[Calling sequence 336](#_bookmark774)

[Import 336](#_bookmark775)

[Export 337](#_bookmark776)

[AEIO\_ModuleInfo 338](#_bookmark777)

[Table 111: AEIO\_ModuleInfo 338](#_bookmark778)

[Behavior Flags 339](#_bookmark779)

[Table 112: AEIO\_ModuleFlags 339](#_bookmark780)

[AEIO\_ModuleFlags2 341](#_bookmark781)

[Table 113: AEIO\_ModuleFlags2 341](#_bookmark782)

[New Kids on the Function Block 342](#_bookmark783)

[Table 114: AEIO\_FunctionBlock4 342](#_bookmark784)

[What Goes In 354](#_bookmark806)

Table 115: [AEGP\_IOInSuite5 354](#_bookmark808)

[What Goes Out 360](#_bookmark811)

[Table 116: AEGPIOOutSuite4 360](#_bookmark812)

[Implementation Details 364](#_bookmark814)

[Export Bit-Depth 364](#_bookmark815)

[User Data vs. Options 364](#_bookmark816)

[CHAPTER 10: PREMIERE PRO 365](#_bookmark817)

[Plug-in installation 366](#_bookmark819)

[Plug-ins... Reloaded 366](#_bookmark820)

[Basic Host Differences 366](#_bookmark821)

[Time 366](#_bookmark822)

[Frame Dimensions 367](#_bookmark823)

[PF\_InData 367](#_bookmark824)

[Parameter UI 367](#_bookmark825)

[Missing Suites 368](#_bookmark826)

[A Special Suite for AE effects Running in Premiere Pro 368](#_bookmark827)

[Multithreading 368](#_bookmark828)

[Bigger Differences 369](#_bookmark830)

[Pixel Formats 369](#_bookmark831)

[32-bit float support 370](#_bookmark832)

[PF\_CHECKOUT\_PARAM and Pixel Formats 370](#_bookmark833)

[Effects presets 370](#_bookmark835)

[Custom ECW UI over a Standard Data Type 371](#_bookmark836)

[Premiere Elements 371](#_bookmark837)

[Unsupported features 371](#_bookmark838)

[But...why’d you LOAD it, if you can’t RUN it?! 372](#_bookmark839)

[Other Hosts? 372](#_bookmark840)

[Reality Sandwich 372](#_bookmark841)

[Function and Suite Reference 373](#_bookmark842)

[General Reference 376](#_bookmark843)

1 : INTRODUCTION

Welcome to the Adobe® After Effects® CC 2015 Software Development Kit! This is a living document, and is constantly being updated and edited. The latest public version of the SDK is available at: <http://www.adobe.com/devnet/aftereffects/>

While we’ve tried to organize this document in a logical order and provide plenty of cross references, your specific needs may vary. Searching through this document based on

keywords will often lead you to your answer. If you need more information, your question may already be answered on the After Effects SDK forum:

<http://forums.adobe.com/community/aftereffects_general_discussion/aftereffects_sdk>.

Use the search box there, and post a new question if your question hasn’t already been answered.

### WHAT CAN I DO WITH THIS SDK?

This SDK describes the Application Programming Interface (API) that developers use to build plug-ins. These plug-ins can extend the capabilities of After Effects and other

applications that support the After Effects API. Plug-ins may also be used to bridge the gap between After Effects and another application.

**WHAT PLUG-INS CAN I BUILD WITH THIS SDK?**

*Effect plug-ins* can be applied to video or audio in a composition, to process video and/or audio data. Some examples of built-in effects are Brightness and Contast, Hue/Saturation, Gaussian Blur, and Warp Stabilizer. Effect plug-ins can provide a set of parameter controls for the user to fine-tune the effect. These parameter values can vary over time, and effects

may use other layers and parameters at different times to calculate the output. Oftentimes, it is thought that plug-ins = effects. But effects are just one type of plug-in used by After

Effects.

*After Effects General Plug-ins (AEGPs)* can read and modify nearly every element of After Effects projects and preferences. They can add menu items, ‘hook’ (register themselves to receive) and trigger After Effects’ internal commands, and add new panels that dock and

resize within the After Effects UI. They can work with markers and keyframes, and manage the render queue. They can even run scripts. Some examples of built-in AEGPs are the AAF importer, and the SWF exporter. Automatic Duck Pro Import AE is another well-known

AEGP.

*After Effects Input/Output (AEIO) plug-ins* provide support for new media file types. Unless you need a custom setup dialog to specify interpretation settings, the [Premiere Pro importer](#_bookmark17)  [API](#_bookmark17) provides similar functionality, and is preferable in many cases. AEIOs use the AEGP

API along with certain APIs specific to AEIOs. While After Effects still supports Photoshop format plug-ins and filters, as well as Foreign Project Format (FPF) plug-ins, these APIs

have been long deprecated in favor of the AEIO API.

*BlitHook* plug-ins output video to external hardware for broadcast quality monitoring and playback to tape. The EMP sample project provides a starting point. In After Effects CC 2014 and later, [Mercury Transmit](#_bookmark18) is the recommended API.

*Artisans* provide rendered output of 3D layers, taking over 3D rendering from After Effects (which still handles all rendering of 2D layers). Artisans use the AEGP API along with

certain APIs specific to Artisans.

Didn’t see the type of integration you need described above? After Effects is very flexible, and there are several [other ways to integrate with After Effects](#_bookmark13).

**WHERE DO PLUG-INS APPEAR IN AFTER EFFECTS?**

Effects plug-ins appear in both the *Effect* menu and the Effects & Presets panel, in the effect category specified in their PiPL. Once they’re applied, the effect’s parameter controls (sliders, pop-ups, etc.) appear in the Effect Controls panel (ECP).

After Effects General Plug-ins (AEGPs) can add items to any After Effects menu, and

additional panels listed in the Window menu. These menu items are indistinguishable from After Effects’ own menu items.

[AEIOs](#_bookmark768) and Photoshop Format plug-ins can appear in the *File > Import* menu, or in the *Import File* dialog in the *Files of type* drop-down, depending on the type of importer. AEIOs and Format plug-ins can also appear as available output formats in the render queue.

BlitHook plug-ins are automatically loaded and used by AE, but do not appear in any menu or dialog. The plug-in may optionally provide a menu item that opens it’s own custom

settings dialog. It would register and update the menu item using the AEGP API. It can registered to be called by After Effects to update the menu with AEGP\_RegisterUpdateMenuHook(), and it can dim/activate the menu item using AEGP\_EnableCommand()/DisableCommand().

Artisans appear in the *Rendering Plug-in* drop-down in the *Advanced* tab of the *Composition Settings* dialog.

**HOW DOES AFTER EFFECTS INTERACT WITH PLUG-INS?**

Plug-ins, written in C or C++, are bundle packages on Mac OS and DLLs on Windows. They must contain a Plug-in Property List ([PiPL](#_bookmark48)) resource on both platforms. The plug-ins must be located in one of a few specific folders in order to be loaded and used by After

Effects.

For effects plug-ins, After Effects sends command selectors (and relevant information) to the plug-in [entry point function](#_bookmark63) designated in the effects’ [PiPL](#_bookmark48) resource. Selectors are sent in response to actions the user takes—applying the effect, changing parameters, scrubbing through frames in the timeline, and rendering all prompt different sequences of selectors.

After Effects creates multiple instances of effects, with settings and input data unique to each sequence. All instances share the same global data, and can share data between all frames

within their sequence. After Effects doesn’t process all image data as soon as the user applies an effect; it invokes effects only when their output is required.

After Effects General Plug-ins (AEGPs) have their entry point function called during

application launch, and register for whatever messaging they need at that time. Further calls to the AEGP are initiated by user actions, as part of the plug-in’s response to menu

commands or UI events. Depending on their features, plug-ins may need to respond to OS- specific entry points as well, for UI work and thread management.

For BlitHook plug-ins, frames are pushed as they're displayed in the Composition panel. Users can initiate a RAM preview on an area of the timeline so that it is rendered to RAM, and then it all gets played out at full speed.

**SDK CONTENTS**

The SDK contains headers defining the After Effects APIs, sample projects demonstrating integration features, and this SDK Guide.

They are compiled with the SDK header files, which expose various After Effects functionality to be used by the plug-in.

### OTHER INTEGRATION POSSIBILITIES

Although this SDK describes the majority of integration possibilities with After Effects, there are other possibilities not to be overlooked.

**SCRIPTING**

Scripting is a relatively nimble and lightweight means to perform automated tasks with After Effects. ScriptUI is one way you can provide UI integration with custom dialogs and panels

(see [HTML5 panels](#_bookmark15) too). And scripting may be used in tandem with plug-in development, in the cases where a certain function is made available via scripting and not via the C APIs

described in this document.

Scripting in After Effects is done using ExtendScript, based on JavaScript. After Effects

includes the ExtendScript ToolKit, a convenient interface for creating and testing your own scripts. Scripts may be compiled into .jsxbin binary files, to protect intellectual property.

You may download the After Effects Scripting Guide, and find a link to the scripting forums, on the Adobe Developer Connection website at: <http://www.adobe.com/devnet/aftereffects/>

After Effects can be driven by executing scripts from the commandline. In your script, you can open the project and run script actions on it. So for example, you can execute the

following statement to run a script from the command line directly:

AfterFX -s "app.quit()"

Or you can execute this statement to run a .jsx script that includes a quit at the end:

AfterFX -r path\_to\_jsx\_script

On Windows, AfterFX.com is the way to get feedback to the console, because AfterFX.com is a command line application.

**HTML5 PANELS**

Starting in CC 2014 and later, After Effects supports HTML5 panels. They are accessed in After Effects from Window > Extensions > (your panel name). Panels can be resized and

docked just like any other panel in After Effects. Panels are built using HTML5, After Effects Scripting, and JavaScript. You may download the After Effects Panel SDK from the the Adobe Developer Connection website at: <http://www.adobe.com/devnet/aftereffects/>

**AERENDER**

Closely coupled with scripting is the command line interface offered by aerender. aerender is primarily suited to allow automated renders, but can be used to execute any sequence of

scripting commands from the command line. An overview is available in the After Effects

help documents here: [http://help.adobe.com/en\_US/aftereffects/cs/using/WS8A8CD670- 4A72-4fb5-AE8E-CB9E232EC0B5a.html](http://help.adobe.com/en_US/aftereffects/cs/using/WS8A8CD670-4A72-4fb5-AE8E-CB9E232EC0B5a.html)

**PREMIERE PRO IMPORTERS**

Premiere Pro importers provide support for importing media into applications across most applications in the Adobe Creative Cloud, including Premiere Pro, Media Encoder, Prelude,

and Audition. Because of this broader compatibility, unless you need very specific integration with After Effects only available via the AEIO API in this SDK, we recommend developing a Premiere Pro importer. The Premiere Pro SDK is available at: [http://www.adobe.com/](http://www.adobe.com/devnet/premiere/)

[devnet/premiere/](http://www.adobe.com/devnet/premiere/)

One advantage of MediaCore importer plug-ins over AEIOs is its priority system: The

highest priority importer gets first crack at importing a file, and if the particular imported

file isn’t supported, the next-highest priority importer will then have the opportunity to try importing it, and so on.

**MERCURY TRANSMIT**

Mercury Transmit plug-ins are used for sending video to output hardware for broadcast-

quality monitoring. Transmitters are support across most applications in the Adobe Creative Cloud, including Premiere Pro, After Effects (starting in CC 2014), Prelude, and

SpeedGrade. The Mercury Transmit API is documented in the Premiere Pro SDK, available at: <http://www.adobe.com/devnet/premiere/>

**SDK AUDIENCE**

You must be a proficient C/C++ programmer to write After Effects plug-ins. While we’ll

help with issues specific to the After Effects API, we can’t help you learn your IDE or basic programming concepts.

This SDK guide assumes you understand After Effects from a user’s perspective, and basic video editing terminology. If you don’t, get the [*Adobe Afte r Effec ts C lassroom in a Book*](http://www.adobe.com/training/books/aftereffects.html), or any of the other fine instructional books on the market. It will help you understand different

color spaces, time-variant parameters, pixel aspect ratio, 3:2 pull-down, alpha channels, and the other subtle After Effects nuances.

**DEVELOPMENT REQUIREMENTS**

The system requirements for After Effects are here:

<http://www.adobe.com/products/aftereffects/systemreqs/>

If you require support for obsolete versions of the application or API, use an old SDK (which we don’t maintain or provide). Six months after the current version is released, we will no

longer provide or support the previous version’s SDK.

The SDK samples are created for XCode 5.0.2 on Mac OS 10.8 or 10.9, and Microsoft Visual Studio .NET 2010 SP 1 (version 10.0) on Windows 7 64. Yes, we’re being pretty stringent about using the required IDE. No, it’s never pleasant to move to a new compiler, but no, we’re not going to continue to help with older build environments.

In order to use Visual Studio, you may need to adjust some installation settings to install the components for compiling 64-bit plug-ins. Visual Studio Express may be used, but will also require an additional installation to compile 64-bit plug-ins, as described here:

<http://msdn.microsoft.com/en-us/library/9yb4317s.aspx>

To compile using newer versions of XCode on newer versions of Mac OS, often all that is required is to update the Base SDK in the Build Settings.

**WHAT’S NEW?**

If this is your first time developing an After Effects plug-in, you can skip the What’s New section and go directly to [How to Start Creating Plug-ins](#_bookmark30).

**WHAT’S NEW IN CC 2015.0.1 (13.5.1)?**

This release fixes some audio APIs that broke in 13.5 due to threading changes. In 13.5, when called on the UI thread, AEGP\_RenderNewItemSoundData() would return A\_Err\_GENERIC. This restores the functionality when called on the UI thread.

To avoid a deadlock, in PF\_Cmd\_UPDATE\_PARAMS\_UI only, AEGP\_RenderNewItemSoundData() will now return silence. This will no longer function as before in this context, but it will continue to work properly elsewhere

**WHAT’S NEW IN CC 2015 (13.5)?**

* Separate UI and Render Threads

This release of After Effects includes major architectural changes to separate the UI (main) thread from the render thread. The render thread sends selectors such as

PF\_Cmd\_RENDER, PF\_Cmd\_SMART\_PRERENDER, and

PF\_Cmd\_SMART\_RENDER to effect plug-ins. The UI thread sends selectors such as PF\_Cmd\_SEQUENCE\_SETUP, PF\_Cmd\_USER\_CHANGED\_PARAM,

PF\_Cmd\_DO\_DIALOG, and PF\_EVENT\_DRAW. PF\_Cmd\_SEQUENCE\_RESETUP is

sent on both render and UI threads.

These changes are to improve interactive performance and responsiveness. At the same time, the new design introduces some new requirements and may break assumptions that existing plug-ins relied on. Here are some of the major changes:

1. The project can no longer be modified by the render thread (and in fact the render thread now has its own local copy of the project)
2. Rendering cannot pass modified sequence data back to the UI thread for custom UI updates
3. In general the UI thread should no longer do time-consuming operations such as synchronously rendering frames

Is your plug-in affected? Test for these problems:

1. Render not updating after UI parameter change because it depends on sequence\_data, which may not be currently copied to render
2. Render not updating during click/drag in the Composition Window (similar reasons)
3. Custom Effect UI not updating because it depends on sequence\_data generated in render (which is no longer available to the UI because it is in a different project, the render project is immutable, and cache contains previously-rendered frames)
4. Errors telling you an operation on the render thread (or UI thread) is not expected

Generally, calculations that will persist or update the UI will now have to be pulled from the UI thread rather than pushed from the render thread. These cases can require use of new

13.5 APIs or different solutions than in past releases.

* The Need For More Efficient Sequence Data Handling

PF\_OutFlag2\_SUPPORTS\_GET\_FLATTENED\_SEQUENCE\_DATA

PF\_Cmd\_GET\_FLATTENED\_SEQUENCE\_DATA

Up to version 13.2, serializing/flattening sequence\_data always involved deallocating and reallocating any data structures. Starting in 13.5, as effect changes are made, serializing/

flattening happens even more often. Why? AE needs to serialize/flatten project changes to send from the UI thread to the render thread, to keep them both synchronized.

To make this process more efficient, starting in 13.5, AE can send

PF\_Cmd\_GET\_FLATTENED\_SEQUENCE\_DATA to request sequence data without requiring the existing data to be deallocated and reallocated. The main difference between this selector and PF\_Cmd\_SEQUENCE\_FLATTEN is that a copy of the correct flattened state is returned without disposing the original structure(s) the effect is currently using. For a code example, refer to the PathMaster sample project.

This will eventually become required for plug-ins that are rebuilt to be thread-safe (see

PF\_OutFlag2\_AE13\_5\_THREADSAFE below). The venerable

PF\_Cmd\_SEQUENCE\_FLATTEN will eventually be unsupported in future versions.

* PF\_OutFlag\_FORCE\_RERENDER Changes

This flag does not work as broadly as before, and we now discourage its usage. Instead we recommend triggering rerenders using one of the following: GuidMixInPtr() (described in the next section), arb data, or PF\_ChangeFlag\_CHANGED\_VALUE. All of these work

better with Undo. Once we have the full set of APIs in place needed to manage render state, we will be able to deprecate FORCE\_RERENDER.

FORCE\_RERENDER doesn’t work in every situation it did before, because it needs to synchronize the UI copy of sequence\_data with the render thread copy.

FORCE\_RERENDER works when set during PF\_Cmd\_USER\_CHANGED\_PARAM. It also works in CLICK and DRAG events, but only if

PF\_Cmd\_GET\_FLATTENED\_SEQUENCE\_DATA is implemented. This is required to prevent flattening and loss of UI state in the middle of mouse operations. Without

GET\_FLATTENED, the new FORCE\_RERENDER behavior will NOT be turned on.

* GUIDs for Cached Frames

PF\_OutFlag2\_I\_MIX\_GUID\_DEPENDENCIES

GuidMixInPtr()

Used by SmartFX only. Use this if custom UI or PF\_Cmd\_DO\_DIALOG changes sequence data, or if the render result depends on anything else not factored in, and rerendering may be needed. During PF\_Cmd\_SMART\_PRERENDER, the effect can call GuidMixInPtr() to mix any additional state that affects the render into our internal GUID for the cached frame. Using this GUID, AE can tell whether the frame already exists or if it needs to be rendered. See an example in SmartyPants sample project.

This is an improvement over the older mechanisms PF\_OutFlag\_FORCE\_RERENDER and PF\_Cmd\_DO\_DIALOG, which would remove the frame from the cache because the host didn't know what else the plug-in was factoring into the rendering. This can also be used rather than PF\_OutFlag2\_OUTPUT\_IS\_WATERMARKED.

* Request Frames Asynchronously Without Blocking the UI

PF\_OutFlag2\_CUSTOM\_UI\_ASYNC\_MANAGER

PF\_GetContextAsyncManager() AEGP\_CheckoutOrRender\_ItemFrame\_AsyncManager() AEGP\_CheckoutOrRender\_LayerFrame\_AsyncManager()

For cases where such renders formerly were triggered by side-effect or cancelled implicity

(such as custom UI histogram drawing), and lifetime is less clear from inside the plug-in, use the new “Async Manager” which can handle multiple simultaneous async requests for effect Custom UI and will automatically support interactions with other AE UI behavior.

Note: Async retrieval of frames is preferred for handling passive drawing situations, but not when a user action will update the project state. If you are (1) responding to a specific user click, AND 2) you need to update the project as a result, the synchronous AEGP\_RenderAndCheckoutLayerFrame() is recommended.

The new HistoGrid sample in the SDK shows how to do completely asynchronous custom UI DRAW event handling on the UI thread when 1 or more frame renders are needed. e.g. for calculating histograms that are shown in the effect pane. Please note there is still a

known bug where drag-changing an upstream param may not refresh the histogram draw until the mouse hovers over it.

* Get Rendered Output of an Effect from its UI

Effects such as keyers or those that draw histograms of post-processed video can retrieve the needed AEGP\_LayerRenderOptionsH using the new function AEGP\_NewFromDownstreamOfEffect() in AEGP\_LayerRenderOptionsSuite. This

function may only be called from the UI thread.

* AEGP Usage on Render Thread

We've tightened validation of when AEGP calls could be used dangerously (such as from the wrong thread or making a change to the project state in render). You may see new errors if code is hitting such cases. For example, making these calls on the render thread will result in an error:

suites.UtilitySuite5()->AEGP\_StartUndoGroup() suites.StreamSuite2()->AEGP\_GetStreamName() suites.StreamSuite2()->AEGP\_SetExpressionState() suites.StreamSuite2()->AEGP\_SetExpression() suites.StreamSuite2()->AEGP\_GetNewLayerStream() suites.StreamSuite2()->AEGP\_DisposeStream() suites.EffectSuite3()->AEGP\_DisposeEffect() suites.UtilitySuite5()->AEGP\_EndUndoGroup()

The solution is to move these calls to the UI thread. Selectors for passive UI updates (such as PF\_EVENT\_DRAW) are not a place to make changes to project state.

Another example of more strict requirements is AEGP\_RegisterWithAEGP(). The documentation has always noted that this function must be called on

PF\_Cmd\_GLOBAL\_SETUP. However in previous versions, plug-ins were able to call this function at other times without running into trouble. Not anymore in 13.5! Calling this

function at other times can cause crashes!

* PF\_Cmd\_SEQUENCE\_RESETUP Called on UI or Render Thread?

There is now a PF\_InFlag\_PROJECT\_IS\_RENDER\_ONLY flag that is only valid in

PF\_Cmd\_SEQUENCE\_RESETUP that will tell you if the effect instance is for render-only purposes. If so, the project should be treated as completely read-only, and you will not be receiving UI related selectors on that effect instance. This can be used to optimize away any UI-only initialization that render does not need. If this flag is false, you should setup UI as normal. This should not be used to avoid reporting errors in render. Errors in render should be reported as usual via existing SDK mechanisms.

* Changes to Avoid Deadlocks

During development, it was noticed that deadlocks could occur in specific call usage. Seatbelts have been introduced to avoid this. The cases occur in

PF\_Cmd\_UPDATE\_PARAMS\_UI when using particular calls because of deprecated synchronous behavior in these calls when used in the UI:

In PF\_Cmd\_UPDATE\_PARAMS\_UI only, PF\_PARAM\_CHECKOUT() for layer parameters will behave as before except that it will return a black frame of the same size, etc., rather than actual rendered pixels. Code that used this for enable/disable detection of parameters

should still work as before. Code that used this for getting analysis frames, etc. outside of PF\_Cmd\_UPDATE\_PARAMS\_UI will work as before.

In PF\_Cmd\_UPDATE\_PARAMS\_UI only, PF\_GetCurrentState() will now return a random GUID. This will no longer function as before in this context, but it will continue to work properly elsewhere.

The above uses should be rare, but if this affects you please contact us about workarounds.

* Deprecated

AEGP\_RenderAndCheckoutFrame() (on the UI Thread). This call should generally not be used on the UI thread since synchronous renders block interactivity.

Use in the render thread is fine. The one case where this may still be useful on the UI thread is a case like a UI button that requires a frame to calculate a parameter which then updates the AE project.

For example, an “Auto Color” button that takes a frame and then adjusts effect params as a result.

A beta of a progress dialog for this blocking operation if it is slow has been implemented, but using this call on the UI thread should be limited to this special cases. The dialog design is

not final.

* Flag for Thread-Safe Effects

PF\_OutFlag2\_AE13\_5\_THREADSAFE

Plug-ins updated for threading should use this flag to tell AE that the plug-in is expected to be UI thread <> Render thread safe.

This flag tells AE that different threads on different AE project copies can be in the effect at the same time but not accessing the same instance. While multiple render threads are not yet in use, this will be useful in future releases.

* Support for Effect Version greater than 7 (new max is MAJOR version 127)

Effects greater than version 7 will now report properly in 13.5 if built with the current SDK headers. It is possible to use these recompiled effects in AE versions older than 13.5, but

internally the version number will wrap modulo 8 (e.g. AE will internally see effect version 8 as version 0).

This can affect the version shown in error dialog display by older AE and affect usage reporting.

Since many older plug-ins were made unloadable in AE with the shift to 64-bit, it should be unlikely this wrapping would cause ambiguity with actual plug-ins in current use (unless these plug-ins have been rapidly increasing version number over the last few years).

However, building with an older SDK and using an 8 or higher version will result in the plug-in reporting an incorrect version to AE, which will then cause mismatch with the PiPL version check for the effect which will have the higher bits set. This is not supported.

If built with an older SDK, you will need to keep the effect version at 7 or below. Increase in version max has been accomplished by adding 4 new higher significant bits to the version that only AE 13.5 and above “sees”. These new high version bits are not contiguous with the

original, preexisting MAJOR version bits -- just ignore the intermediate bits. The new version layout looks like this in hexadecimal or binary.

0x 3C38 0000

^^ original MAJOR version bits as a hex mask 0-7

^^ new HIGH bits extending the original MAJOR version bits 8-127

0b 0011 1100 0011 1000 0000 0000 0000 0000

^^ ^ original MAJOR version bits as a hex mask 0-7

^^ ^^ ignore / do not use

^^ ^^ new HIGH bits extend the original MAJOR version bits 8-127.

These bits are ignored in AE versions older than 13.5.

* New Installer Hints for MacOS

Developers can find paths to the default location of plug-ins, scripts, and presets on Mac OS X in a new plist file (same as the paths in the Windows registry): /Library/Preferences/ com.Adobe.After Effects.paths.plist

You can use the values in this plist to direct where your installers or scripts write files, in the same way that you would use the paths keys in the registry on Windows:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Adobe\After Effects\13.5

* Work In Progress

AEGP\_RenderAndCheckoutLayerFrame\_Async() AEGP\_CancelAsyncRequest()

This APIs are in progress, and should not be used yet.

**WHAT’S NEW IN CC 2014.1 (13.1)?**

PF\_CreateNewAppProgressDialog()

It won't open the dialog unless it detects a slow render. (2 seconds timeout).

**WHAT’S NEW IN CC 2014 (13.0)?**

Starting in CC 2014, After Effects will now honor a change to a custom UI height made using [PF\_UpdateParamUI](#_bookmark320).

[AEGP Effect Suite](#_bookmark607) is now at version 4, adding new functions to work with effect masks. AEGP\_RenderSuite is now at version 4, adding a new function

[AEGP\_RenderAndCheckoutLayerFrame](#_bookmark686), which allows frame checkout of the current layer with effects applied at non-render time. This is useful for an operation that requires the frame, for example, when a button is clicked and it is acceptable to wait for a moment while it

is rendering. Note: Since it is not asynchronous, it will not solve the general problem where custom UI needs to draw based on the frame. The layer render options are specified using the new [AEGP\_LayerRenderOptionsSuite](#_bookmark683).

[Mercury Transmit](#_bookmark18) plug-ins and [HTML5 Panels](#_bookmark15) are now supported.

**WHAT’S NEW IN CC (12.0)?**

Effect names can now be up to 47 characters long, up from 31 characters previously.

We added the [PF\_AngleParamSuite](#_bookmark311), providing a way to get floating point values for angle parameters. [PF App Suite](#_bookmark349) version 5 adds [PF\_AppGetLanguage](#_bookmark351) to query the current

language so that a plug-in can use the correct language string, as well as several new

PF\_App\_ColorType enum values for new elements whose colors can be queried.

[AEGP Persistent Data Suite](#_bookmark673) is now at version 4, adding a new parameter to AEGP\_GetApplicationBlob to choose between retrieving several different application blobs. There are also new functions to get/set time and ARGB values.

[AEGP\_CompSuite](#_bookmark576) is now at version 10, adding new functions to check/modify whether

layer names or source names are shown, and whether the blend modes column is shown or not. Also added are new functions to get and set the Motion Blur Adaptive Sample Limit.

[AEGP\_LayerSuite](#_bookmark591) is now at version 8, adding new functions to set/get the layer sampling quality. AEGP\_CanvasSuite is also now at version 8. The new function [AEGP\_MapCompToLayerTime](#_bookmark746) handles time remapping with collapsed or nested comps, unlike AEGP\_ConvertCompToLayerTime.

AEGP\_UtilitySuite is now at version 6, adding a new Unicode-aware function: [AEGP\_ReportInfoUnicode](#_bookmark667). Another new function, [AEGP\_GetPluginPaths](#_bookmark671), provides some useful paths related to the plug-in and the After Effects executable itself.

The behavior for [AEGP\_NewPlaceholderFootageWithPath](#_bookmark585) has been updated, so that the file\_type should now be properly set, otherwise a warning will appear.

[AEGP\_InsertMenuCommand](#_bookmark554) can now insert menu items in the File>New submenu.

[AEGP\_I OInSuite](#_bookmark807) is now at version 5, adding new functions to get/set/clear the native start time, and to get/set the drop-frame setting of footage.

**WHAT’S NEW IN CS6.0.1 (11.0.1)?**

New in 11.0.1, the AE effect API version has been incremented to 13.3. This allows effects to distinguish between 11.0 and 11.0.1. There is a bug in 11.0 with the Global Performance

Cache, when a SmartFX effect uses both [PF\_OutFlag2\_AUTOMATIC\_WIDE\_TIME\_INPUT](#_bookmark178)

& [PF\_OutFlag\_NON\_PARAM\_VARY](#_bookmark147). Calling [checkout\_layer](#_bookmark408) during

PF\_Cmd\_SMART\_PRE\_RENDER returns empty rects in PF\_CheckoutResult. The

workaround is to simply make the call again. This workaround is no longer needed in 11.0.1.

**WHAT’S NEW IN CS6 (11.0)?**

We’ve made several refinements for better parameter UI handling. [PF\_PUI\_INVISIBLE](#_bookmark215) parameter UI flag is now supported in After Effects, which is useful if your plug-in needs hidden parameters that affect rendering. Now when a plug-in disables a parameter using [PF\_UpdateParamUI](#_bookmark320), we now save that state in the UI flags so that the plug-in can check the flag in the future to see if it is disabled. A new flag,

[PF\_ParamFlag\_SKIP\_REVEAL\_WHEN\_UNHIDDEN](#_bookmark224), allows a parameter to be unhidden

without twirling open any parents and without scrolling the parameter into view in the Effect Controls panel and the Timeline panel.

Effects that render a watermark over the output when the plug-in is in trial mode can now tell After Effects whether watermark rendering mode is on or off, using the new [PF\_OutFlag2\_OUTPUT\_IS\_WATERMARKED](#_bookmark182).

The new Global Performance Cache means you must tell After Effects to discard old cached frames [when changing your effect’s rendering](#_bookmark394).

We’ve removed PF\_HasParamChanged and PF\_HaveInputsChangedOverTimeSpan, providing [PF\_AreStatesIdentical](#_bookmark324) instead.

Effects that provide custom UI can now receive [PF\_Event\_MOUSE\_EXITED](#_bookmark433), to gain

notification that the mouse exited the layer or comp panel. [PF\_ParamUtilsSuite](#_bookmark318) is now at version 3.

[PF\_GET\_PLATFORM\_DATA](#_bookmark500) now has new selectors for getting the wide character path of the executable and resource file: PF\_PlatData\_EXE\_FILE\_PATH\_W and PF\_PlatData\_RES\_FILE\_PATH\_W. The previous non-wide selectors are now deprecated.

3D is a major theme of AE CS6. A new AEGP\_LayerFlag\_ENVIRONMENT\_LAYER has been added. Many new [layer streams](#_bookmark623) were added. Additionally, AEGP\_LayerStream\_SPECULAR\_COEFF was renamed to AEGP\_LayerStream\_SPECULAR\_INTENSITY, AEGP\_LayerStream\_SHININESS\_COEFF was

renamed to AEGP\_LayerStream\_SPECULAR\_SHININESS, and

AEGP\_LayerStream\_METAL\_COEFF was renamed to just AEGP\_LayerStream\_METAL.

A new suite, [AEGP\_RenderQueueMonitorSuite](#_bookmark704), provides all the info a render queue manager needs to figure out what is happening at any point in a render.

[AEGP Mask Suite](#_bookmark652) is now at version 6, and provides functions to get and set the mask

feather falloff type. [AEGP Mask Outline Suite](#_bookmark658) is now at version 3, and provides access to get and set mask outline feather information.

Effects that depend on masks now have a new flag available,

[PF\_OutFlag2\_DEPENDS\_ON\_UNREFERENCED\_MASKS](#_bookmark180).

[AEGP Comp Suite](#_bookmark576) is now at version 9. AEGP\_CreateTextLayerInComp and

AEGP\_CreateBoxTextLayerInComp now have a new parameter, select\_new\_layerB.

[AEGP Render Suite](#_bookmark685) is now at version 3, adding a new function to get the GUID for a render receipt.

Finally, we have added two new read-only [Dynamic Stream](#_bookmark636) flags: AEGP\_DynStreamFlag\_SHOWN\_WHEN\_EMPTY and AEGP\_DynStreamFlag\_SKIP\_REVEAL\_WHEN\_UNHIDDEN.

For effects running in Premiere Pro CS6, we have added the ability to get 32-bit float and YUV frames from [PF\_CHECKOUT\_PARAM](#_bookmark834).

**...AND WHAT WAS NEW BEFORE CS6?**

For history this far back, see obsolete copies of the SDK (which we don’t provide; if someone wants you do develop for antique software, they’d best provide the SDK).

### HOW TO START CREATING PLUG-INS

**PLAY!**

Before you write a line of code, Spend some significant time playing with After Effects, and with the [sample projects](#_bookmark38). [Build the plug-ins](#_bookmark40) into the right folder. Set lots of breakpoints, read the amusing and informative comments.

**PLAN!**

Be clear on what your plug-in will attempt to do.

**HACK!**

After experimenting with the samples, find one that does something *like* what you want to

do. The temptation to start from scratch may be strong; fight it! For effects, use the Skeleton template project. Avoid the headache of reconstructing projects (including the troublesome

custom build steps for Windows PiPL resource generation) by grafting your code into an existing project.

**STEAL!**

To make the Skeleton sample your own, copy the entire \Skeleton directory, renaming it to (for example) \WhizBang. Using your text editor of choice, search \WhizBang\\*.\* (yes, that includes .NET and XCode project files) for occurrences of Skeleton and SKELETON, and replace them with WhizBang and WHIZBANG.

You now have a compiling and running plug-in that responds to common commands, handles 8 and 16-bpc color, uses our AEGP\_SuiteHandler utility code, and responds to 3D light and camera information. There, was that so hard?

AEGP developers will do well to start with Projector (for After Effects project creation support), Easy Cheese for a keyframe assistant, IO for media file format support, and Persisto for a simple menu command and working with preferences.

**TEST!**

If only for testing convenience, you should have a project saved with your effect applied, and all its parameters keyframed to strange values. Between these projects which stress your plug- in, and the tools provided by your development environment, you’re well on your way to

shipping some tested code.

**BLAME!**

If you run into behavior that seems wrong, see if you can reproduce the behavior using one of the unmodified sample projects. This can save you a lot of time, if you can determine whether the bug behavior was introduced by your modifications, or was already there to begin with.

**DEVELOPERS MATTER**

Third party developers drive API and SDK improvement and expansion. Your products enable After Effects to do things we’d never considered. Your efforts make After Effects better; keep it up!

We work hard on the SDK, and welcome your comments and feedback. Almost every change we make to the API is suggested by developers like you. [Talk to us](mailto:zlam@adobe.com).

### SAMPLE PROJECTS

There is at least one sample of every type of plug-in supported by the current API, as well as projects to illustrate particular concepts.

In the sample projects, we’ve kept the code as simple as possible. A showy implementation might get us good grades in a programming class, but won’t help you understand how to use API features.

After the break, we explain how to build the sample projects, so keep reading below!

**TABLE 1: SAMPLE PROJECT DESCRIPTIONS**

|  |  |
| --- | --- |
| **Project** | **Description** |
| AEGPs | AEGPs hook directly into After Effects’ menus and other areas in the UI. See below for specifics on where the AEGP appears in the UI. |
| Artie | Artie the Artisan takes over rendering of all 3D layers in a given composition. This is the same API used by our internal 3D renderers; it is very complex, and exposes a great deal of tacit information about the After Effects rendering pipeline. Unless you have a compelling reason to replace the way After Effects handles 3D rendering, you need never work with this sample. Artisans appear in Composition > Composition Settings, in the Advanced tab, in the Rendering Plug-in drop-down. |
| Easy Cheese | A keyframer (which shows up on the Animation > Keyframe Assistant submenu), Easy Cheese shows how to manipulate various characteristics of keyframes (in a way that, uncannily, resembles our shipping plug-in, Easy Ease...) |
| FBIO | Exercises the After Effects Input/Output (AEIO) API. Similar to the IO sample, but supports the frame-based .ffk file format. Note that we now recommend developing a [*Premiere Pro importer*](#_bookmark17) instead. |
| Grabba | Gets frames (formatted as the plug-in requests) from any composition in the project. |
| IO | Exercises the After Effects Input/Output (AEIO) API. Supports the fictitious  .fak file format, and handles all requests from After Effects for retrieving data from or outputting to such files. Note that we now recommend developing a  [*Premiere Pro importer*](#_bookmark17) instead. |
| Mangler | Mangler is a keyframer demonstrating the use of an ADM palette, just like our own. |

|  |  |
| --- | --- |
| **Project** | **Description** |
| Panelator | Creates a panel that can be docked along with the rest of the standard panels. Note: It is far more work to create a panel this way than using the HTML5 Panel SDK. We recommend starting with that SDK instead. |
| Persisto | Shows how to read and write information from the After Effects preferences file. |
| ProjDumper | Creates a text file representing every element in an After Effects project. |
| Projector | Imports the (fictitious) .sdk file format, and creates a project using AEGP API calls. Whenever you’re wondering how to get or set some characteristic of a project element, look here first. Note: There are some hardcoded paths in Projector.h. If you don't set these to refer to actual media on disk, you WILL get errors while running this plug-in. Don't blame us; change them! |
| QueueBert | Pronounced “Cue-BARE!”, QueueBert manipulates all aspects of render queue items and the output modules associated with them. |
| Streamie | Manipulates streams, both dynamic and fixed. |
| Sweetie | Sweetie uses the PICA (or “Suite Pea”) API to provide a function Suite, for use by other plug-ins. If you’re writing multiple plug-ins that rely on the same image processing library, you could provide the library functionality using such a suite. |
| Text Twiddler | Manipulates text layers and their contents. |
| Effects | All effects appear in the Effects & Presets panel, and in the Effect menu. |
| Checkout | Checks out (of After Effects’ frame cache) a frame of input from another layer, at a specified time. This is an important concept for all effects with layer parameters. Premiere Pro compatible. |
| Convolutrix | Exercises our image convolution callbacks. Premiere Pro compatible. |
| Gamma Table | Shows how to manage sequence data, and uses our iteration callbacks. For nostalgia’s sake, we’re leaving this one sample in C; it’s also compatible with many third-party plug-in hosts, due to its reliance on version 3.x API features. |
| Paramarama | Exercises wayward param types not used in other sample. Premiere Pro compatible. |
| PathMaster | Shows how to access paths from within an effect. |
| Portable | Shows how to detect and respond to several different plug-in hosts. Premiere Pro compatible. |
| Resizer | Resizer resizes (surprise!) the output buffer. This is useful for effects like glows and drop shadows, which would be truncated at the layer’s edges if they didn’t expand the output buffer. Premiere Pro compatible. |
| SDK Backwards | Reverses a layer’s audio, and mixes it with a keyframe-able sine wave. |
| SDK Noise | Premiere Pro compatible, demonstrates 32-bit and YUV rendering in Premiere Pro. |

|  |  |
| --- | --- |
| **Project** | **Description** |
| Shifter | Shifts an image in the output buffer, and exercises our transform\_world and subpixel sampling functions. |
| SmartyPants | Demonstrates the SmartFX API, required for support of floating point pixels. |
| Transformer | Exercises our image transformation callbacks. |
| Effect Template | |
| Skeleton | Skeleton is the starting point for developing effects. Premiere Pro compatible. |
| Effects with Custom UI | |
| CCU | Implements a custom user interface in the composition and layer windows, supporting pixel aspect ratio and downsample ratios. Premiere Pro compatible. |
| ColorGrid | Shows how to use arbitrary data type parameters. Also has a nice custom UI. Premiere Pro compatible. |
| Custom ECW UI | Implements a very boring custom user interface in the effect controls window, and shows how to respond to numerous UI events. |
| Histogrid | New for CC 2015 (13.5). An example of how custom UI can access asynchronously-rendered upstream frames for lightweight processing in CC 2015 and later. This effect calculates a sampled 10x10 color grid from the upstream frame, and displays a preview of that color grid. In render, a higher- quality grid is calculated and used to modify the output image, creating a blend of a color grid with the original image. |
| Supervisor | Shows how to control parameters (both values and UI) based on the value of other parameters. Premiere Pro compatible. |
| BlitHook | |
| EMP | External Monitor Preview. Use this as a starting point for adding support to output video from the composition panel to video hardware. |

**BUILDING THE SAMPLE PROJECTS**

We’ve combined the sample projects into a single master project, stored in the Examples folder of the SDK. For Mac OS, it is Buildall.xcodeproj; for Windows, it is BuildAll.sln.

In your IDE, you’ll need to change the output folder of your project to build into After

Effects’ plug-in folder. For development, we recommend using the following path for Mac OS:

/Library/Application Support/Adobe/Common/Plug-ins/[version]/MediaCore/

Version is locked at 7.0 for all CC versions, or CSx for earlier versions.

for example: /Library/Application Support/Adobe/Common/Plug-ins/7.0/MediaCore/

or: /Library/Application Support/Adobe/Common/Plug-ins/CS6/MediaCore/

and the following path for Windows:

[Program Files]\Adobe\Common\Plug-ins\[version]\MediaCore\

for example: C:\Program Files\Adobe\Common\Plug-ins\7.0\MediaCore\

or: C:\Program Files\Adobe\Common\Plug-ins\CS6\MediaCore\

Note that this Windows path is only recommended for development purposes. Windows installers should follow the guidelines [here](#_bookmark56).

In Xcode, you can set this path once for all projects in the Xcode Preferences > Locations > Derived Data > Advanced. Under *Build Location* choose *Custom*, and fill in the path.

In Visual Studio, for convenience, we have specified the output path for all sample projects using the environment variable AE\_PLUGIN\_BUILD\_DIR. You’ll need to set this as a user

environment variable for your system. On Windows 7, right-click *My Computer* > *Properties*

>and in the left sidebar choose *Advanced System Settings*. In the new dialog, hit the

*Environment Variables* button. In the User variables area, create a New variable named AE\_PLUGIN\_BUILD\_DIR, and with the path described above. Log out of Windows and log back in so that the variable will be set.

Alternatively, you can set output path for each project individually in Visual Studio by right- clicking a project in the Solution Explorer, choosing Properties, and then in Configuration Properties > Linker > General, set the Output File.

### DEBUGGING PLUG-INS

The best way to learn the interaction(s) between After Effects and plug-ins is running the samples in your debugger. Spending some quality time in your compiler’s debugger, and a sample project that closely resembles your plug-in, can really pay off.

Specify After Effects as the application to run during debug sessions.

On Windows, in the Visual Studio solution, in the Solution Explorer panel, right-click on the project, and choose Properties. In Configuration Properties > Debugging > Command, provide the path to the After Effects executable file.

For Mac OS, in the XCode project, in the Groups and Files panel, in the Executables section, create a New Custom Executable. Set the Executable Path to the After Effects executable file.

**DELETING PREFERENCES**

During the course of developing a plug-in, your plug-in may pass settings information to After Effects, which is then stored in its preferences file. You may delete the preferences and restart After Effects with a clean slate by holding down Ctrl-Alt-Shift / Cmd-Opt-Shift

during launch.

On Windows, the preferences are stored here:

[user folder]\AppData\Roaming\Adobe\After Effects\[version]\Adobe After Effects [version]-x64 Prefs.txt

On Mac OS, they are stored here:

~/Library/Preferences/Adobe/After Effects/[version]/Adobe After Effects [version]-x64 Prefs

### COMPATIBILITY ACROSS MULTIPLE VERSIONS?

Generally, you should compile your plug-ins with the latest After Effects SDK headers. This makes the latest suites and API functionality available to your plug-ins. When a new version of After Effects is released, you generally will not need to provide a new version unless you wish to take advantage of new functionality exposed through the new SDK. However, you

should always test your plug-in in new versions of After Effects before claiming compatibility with new versions.

You should test your plug-in thoroughly in each version of After Effects supported by your plug-in. If you need to add a conditional block of code to be run only in specific versions of After Effects, you can always check the API version in [PF\_InData](#_bookmark111).version for effects, or in the major and minor\_versionL passed into your AEGP in the [EntryPointFunc()](#_bookmark539).

For even more precise version checking, a plug-in can run a script using [AEGP\_ExecuteScript](#_bookmark670), querying one of the following attributes: app.version - e.g. 11.0.1x12

app.buildNumber - e.g. 12.

**TABLE 2: API VERSIONS**

|  |  |  |
| --- | --- | --- |
| **Release** | **Effect API Version** | **AEGP API Version** |
| CC 2015 (13.5, 13.5.1) | 13.9 | 113.5 |
| CC 2014 (13.0-13.2) | 13.7 | 113 |
| CC (12.2) | 13.6 | 112.2 |
| CC (12.1) | 13.5 | 112.1 |

**TABLE 2: API VERSIONS**

|  |  |  |
| --- | --- | --- |
| **Release** | **Effect API Version** | **AEGP API Version** |
| CC (12.0) | 13.4 | 112.0 |
| CS6.0.1 (11.0.1) | 13.3 | 111.0 |
| CS6 (11.0) | 13.2 | 111.0 |
| CS5.5 (10.5) | 13.1 | 17.0 |
| CS5 (10.0) | 13.0 | 17.0 |
| CS4 (9.0) | 12.14 | 16.24 |
| CS3 (8.0) | 12.13 | 16.24 |
| 7.0 | 12.12 |  |
| 6.5, 6.0 | 12.10 (Check for the presence of updated AEGP suites, should you need to differentiate between 6.0 and 6.5.) |  |
| 5.0 | 12.5 |  |
| 4.1 | 12.2 |  |
| 3.1 | 11.6 |  |

### THIRD-PARTY PLUG-IN HOSTS?

Some developers are wary of using each After Effects release’s new API features, to maintain compatibility with hosts with partial implementations. You can distinguish between host

applications by checking [PF\_InData](#_bookmark111)>appl\_id. After Effects uses the appl\_id ‘FXTC’.

Premiere Pro uses ‘PrMr’. As of this writing, no third party hosts support SmartFX, or our AEGP functions. Also, see the chapter on compatibility with [Premiere Pro and other hosts](#_bookmark818).

**PIPL RESOURCES**

Originating in Adobe Photoshop over a decade ago, Plug-In Property Lists, or PiPLs, are resources which provide basic information about a plug-in’s behavior, without executing the plug-in. PiPLs have been largely supplanted within After Effects by

[*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark74) and dynamic outflags. However, for archaeological reasons, the behaviors indicated during PF\_Cmd\_GLOBAL\_SETUP must agree with those in the PiPL.

A PiPL specifies the entry point of a plug-in, the display name, as well as the plug-in’s match name. The match name is a unique, constant identifier, unlike a plug-in’s display name, which may be changed dynamically. Starting in CC, display names can be up to 47 characters long. Previously, they were limited to 31 characters.

In the interest of cross-platform compatibility, use a single .r file for both Mac OS and Windows versions of your plug-in, like the samples do. PiPL properties must always be in

Mac OS-specific byte order. On Windows, PiPLs are compiled by processing a .r file through

pipltool.exe, which converts the .r file into a binary .rc file. The Windows sample

projects all contain custom build steps which generate a .rc file, using a cross-platform .r file and our cnvtpipl.exe command line utility. Base your development on an existing sample plug-in and the build step will be correctly implemented.

**ENTRY POINT**

Your plug-in’s entry point is exported through the PiPL on Windows and Mac OS. If the plug-in supports multiple platforms (e.g. Windows and Intel Macs), then multiple entry points must be defined in the PiPL. There is no need for a Windows .def file or manual exports, unless you’re also designating some other OS-specific entry point. The macros

defined in entry.h (in the \SDK\Examples\Headers directory) take care of exporting each sample’s entry point function. XCode seems overly concerned about the prospect of a main() function returning a long; all the sample projects’ entry point functions have been changed to the seemingly innocuous EntryPointFunc().

**PIPL RESOURCES AND MICROSOFT VISUAL STUDIO**

To use resources from Microsoft Visual Studio .NET with pipltool-generated resources,

#include the output of the custom build steps into the Microsoft-generated .rc file.

// in file WhizBang.rc, generated by .NET.

#include “WhizBang\_PiPL\_temp.rc” // pipltool.exe’s output

If modifying a sample plug-in, change the name of the file generated by pipltool.exe to

something like WhizBang\_PiPL\_temp.rc, or it will overwrite the Microsoft resources each time you build; not good.

**MULTIPLE PIPLS**

It is possible, but not recommended, to include multiple plug-ins (both AEGPs and effects) in the same file, using multiple PiPLs. If there are PiPLs for both AEGPs and effects in the same file, the AEGPs must come first!

No other hosts (not even Premiere Pro) support multiple PiPLs pointing to multiple effects within the same .dll or code fragment. Also, if you need to update one plug-in, do you really want to ship a new build of all your plug-ins? We recommend one PiPL, and one plug-in, per code fragment.

**SUPER SECRET PIPL BIT**

For those of you who use C++ and simply *must* keep your plug-ins loaded all the time (to avoid having your v-tables trashed, among other hazards), set the PiPL’s AE\_Reserved\_Info member to 8. Over the years we’ve been quite stringent, insisting that plug-ins be good memory citizens and respond gracefully to getting unloaded. We know there are cases in which being unloaded with no warning can really ruin a plug-in’s day (and v-

tables), and so have provided this work-around. Be nice, perform scrupulous memory management, and only use your powers for good.

**WHY DO I NEED TO KNOW ALL THIS?**

You don’t; After Effects does. If you follow our advice and base your projects on the SDK

samples, you can simply change the .r file containing your PiPL definition(s), and your plug- in’s resources will be automagically updated the next time you build. Feel the love. Or, if you ever tinker with the custom build steps, feel the pain.

### EXCEPTIONS

Handle all exceptions generated by your plug-in’s code, *within* your plug-in. Pass those which didn’t originate in your plug-in’s code to After Effects. After Effects’ APIs are

designed for plug-ins written in C, and don’t expect exceptions. After Effects will crash

immediately if one is thrown from within a plug-in. The effect samples use a firewall around the switch statement in the main() function, and the AEGPs wrap their function hooks in try/catch blocks.

## WHERE INSTALLERS SHOULD PUT PLUG-INS

Installing your plug-ins in the common location will allow them to be loaded by Premiere Pro, if installed.

On Windows, the common plug-ins folder can be found (as an explicit path) in the following registry entry:

HKLM\SOFTWARE\Adobe\After Effects\[version]\CommonPluginInstallPath

On Mac, the common plug-ins folder is at:

/Library/Application Support/Adobe/Common/Plug-ins/[version]/MediaCore

Version is locked at 7.0 for all CC versions, or CSx for earlier versions.

for example: /Library/Application Support/Adobe/Common/Plug-ins/7.0/MediaCore/

Do not use Mac OS aliases or Windows shortcuts, as these are not traversed by Premiere Pro.

**DO I HAVE TO INSTALL THE PLUG-INS TO THE COMMON FOLDER?**

You may have good reason to install your plug-in for only After Effects, for example, if your plug-in depends on suites and functionality not available in Premiere Pro. We strongly recommend that you use the common folder whenever possible, but for certain cases, the AE- specific plug-in folder is still available.

On Windows, the app-specific plug-ins folder can be found (as an explicit path) in the following registry entry:

\\HKEY\_LOCAL\_MACHINE\SOFTWARE\Adobe\After Effects\(version)\PluginInstallPath

On Mac OS, the app-specific plug-ins folder is at:

/Applications/Adobe After Effects [version]/Plug-ins/

When launched, After Effects recursively descends 10 levels deep into subdirectories of its path. Mac OS aliases are traversed, but Windows shortcuts are not. Directories terminated by parentheses or preceded by the symbols ¬ (Mac OS) or ~ (Windows) are not scanned.

Try as you might to build a fence between AE and Premiere Pro, users will still find ways to get across using our lovely integration goodness - Your effects will still be available to

Premiere Pro users who create a dynamically linked AE composition with your effect, and put it in a Premiere Pro sequence.

**LOCALIZATION**

Starting in CC, [PF App Suite](#_bookmark349) adds PF\_AppGetLanguage() to query the current language so that a plug-in can use the correct language string.

When passing strings to AE, some parts of the API accept Unicode. In other areas, for

example when specifying effect parameter names during PF\_Cmd\_PARAMS\_SETUP, you'll need to pass the names in a char string. For these non-Unicode strings, AE interprets strings as

being multi-byte encoded using the application's current locale. To build these strings, on Windows you can use the WideCharToMultiByte() function, specifying CP\_OEMCP as the first argument. On Mac OS, use the encoding returned by GetApplicationTextEncoding().

Testing with different languages in AE doesn’t require an OS reinstallation, but it does require a reinstallation of AE:

For Win, change the system locale to the targeted language (control panel > region and

language > administrative tab > change system locale), restart machine, and then install AE in the according language.

For Mac, set targeted language to the primary language in the preferred language list, and then install AE in the according language.

## NEXT STEPS

You now have an understanding of what plug-ins are, what they can do, and how After Effects communicates with them. We will cover the basics of effects plug-ins in the next chapter.

2 : EFFECT BASICS

This chapter will provide all the information you need to know to understand how a basic effect plug-in works. These details are fundamental to every effect plug-in. By the time you finish this chapter, you’ll be ready for the fun stuff; modifying pixels!

### ENTRY POINT

All communication between After Effects and an effect plug-in is initiated by After Effects, and it all happens by the host (After Effects) calling a single entry point function. For all

effect plug-ins, the entry point function must have the following signature:

PF\_Err main (

PF\_Cmd cmd,

PF\_InData \*in\_data, PF\_OutData \*out\_data, PF\_ParamDef \*params[], PF\_LayerDef \*output, void \*extra)

[The name of the entry point function above is “main”, but it can be whatever is specified in](#_bookmark49) [the](#_bookmark49) PiPL resource.

Before each call to the entry point function, After Effects updates [PF\_InData](#_bookmark110) and the plug- in’s parameter array PF\_ParamDef[] (except as noted). After the plug-in returns from its

call, After Effects checks [PF\_OutData](#_bookmark127) for changes and, when appropriate, uses the

PF\_LayerDef the effect has rendered.

**TABLE 3: ENTRY POINT FUNCTION PARAMETERS**

|  |  |
| --- | --- |
| **Argument** | **Purpose** |
| [cmd](#_bookmark72) | After Effects sets the [*command selector*](#_bookmark70) to tell the plug-in what to do. |
| [in\_data](#_bookmark110) | Information about the application’s state and the data the plug-in is being told to act upon. Pointers to numerous interface and image manipulation functions are also provided. |

**TABLE 3: ENTRY POINT FUNCTION PARAMETERS**

|  |  |
| --- | --- |
| **Argument** | **Purpose** |
| [out\_data](#_bookmark127) | Pass back information to After Effects by setting fields within out\_data. |
| [params](#_bookmark184) | An array of the plug-in’s parameters at the time provided in in\_data> current\_time. params[0] is the input image (a [PF\_EffectWorld](#_bookmark226)) to which the effect should be applied. These values are only valid during certain selectors (this is noted in the [*selector*](#_bookmark72)  [*descriptions*](#_bookmark72)). Parameters are discussed at length [*here*](#_bookmark206). |
| [output](#_bookmark226) | The output image, to be rendered by the effect plug-in and passed back to After Effects. Only valid during certain selectors. |
| [extra](#_bookmark435) | The extra parameter varies with the command sent or (in the case of [PF\_Cmd\_EVENT](#_bookmark99)) the [*event type*](#_bookmark420). Used primarily for [*event management*](#_bookmark425)and [*parameter supervision*](#_bookmark314). |

## COMMAND SELECTORS

Commands are, simply, what After Effects wants your effect to do. Responses to some

selectors are required; most are optional, though recall that we did add them for a *reason*...

With each command selector sent, effects receive information from After Effects in

[PF\_InData](#_bookmark110), input and parameter values in PF\_ParamDef[] (an array of parameter

descriptions including the input layer), and access to callbacks and function suites. They send information back to After Effects in [PF\_OutData](#_bookmark127), and (when appropriate) render output to a PF\_LayerDef, also called a [PF\_EffectWorld](#_bookmark226). During events, they receive event-specific information in [extra](#_bookmark435).

**CALLING SEQUENCE**

Only the first few command selectors are predictable; the rest of the calling sequence is dictated by user action.

When first applied, a plug-in receives [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75), then [*PF\_Cmd\_PARAM\_SETUP*](#_bookmark77). Each time the user adds the effect to a layer, [*PF\_Cmd\_SEQUENC E\_SETUP*](#_bookmark79) is sent.

For each frame rendered by a basic non-SmartFX effect, After Effects sends

[*PF\_Cmd\_FRAME\_SETUP*](#_bookmark86), then [*PF\_Cmd\_RENDER*](#_bookmark88), then

[*PF\_Cmd\_FRAME\_SETD OWN*](#_bookmark90). All effect plug-ins must respond to [*PF\_Cmd\_RENDER*](#_bookmark88)*.*

For SmartFX, [*PF\_Cmd\_SMART\_PRE\_RENDER*](#_bookmark95) may be sent any number of times, before a single [*PF\_Cmd\_SMART\_RENDER*](#_bookmark97) is sent.

[*PF\_Cmd\_SEQUENC E\_SETD OWN*](#_bookmark85) is sent on exit, when the user removes an effect or closes the project. [*PF\_Cmd\_SEQUENCE\_RESETUP*](#_bookmark81) is sent when a project is loaded or when the layer to which it’s applied changes. [*PF\_Cmd\_SEQUENCE\_FLAT TEN*](#_bookmark83) is sent when the After Effects project is written out to disk.

[*PF\_Cmd\_ABOU T*](#_bookmark73) is sent when the user chooses *About…* from the Effect Controls Window (ECW).

[*PF\_Cmd\_GLOBAL\_SETD OWN*](#_bookmark76) is sent when After Effects closes, or when the last instance of the effect is removed. Do not rely on this message to determine when your plug-in is being removed from memory; use OS-specific entry points.

**TABLE 4: COMMAND SELECTORS**

|  |  |
| --- | --- |
| **Selector** | **Response** |
| Global Selectors  All plug-ins must respond to these selectors. | |
| PF\_Cmd\_ABOUT | Display a dialog describing the plug-in. Populate out\_data>return\_msg and After Effects will display it in a simple modal dialog. Include your plug- in’s version information in the dialog. On Mac OS, the current resource file will be set to your effects module during this selector. |
| PF\_Cmd\_GLOBAL\_SETUP | Set any required flags and PF\_OutData fields (including out\_data>my\_version) to describe your plug-in’s behavior. |
| PF\_Cmd\_GLOBAL\_SETDOWN | Free all global data (only required if you allocated some). |
| PF\_Cmd\_PARAM\_SETUP | Describe your parameters and register them using [PF\_ADD\_PARAM](#_bookmark283). Also, register custom user interface elements. Set [PF\_OutData](#_bookmark127)>num\_params to match your parameter count. |
| Sequence Selectors  These control sequence data handling. | |
| PF\_Cmd\_SEQUENCE\_SETUP | Allocate and initialize any sequence-specific data. Sent when the effect is first applied. [PF\_InData](#_bookmark110) is initialized at this time. |

|  |  |
| --- | --- |
| **Selector** | **Response** |
| PF\_Cmd\_SEQUENCE\_RESETUP | Re-create (usually unflatten) sequence data. Sent after sequence data is read from disk, during pre-composition, or when the effect is copied; After Effects flattens sequence data before duplication. During duplication, PF\_Cmd\_SEQUENCE\_RESETUP is sent for both the old and new sequences. Don’t expect a PF\_Cmd\_SEQUENCE\_FLATTEN between PF\_Cmd\_SEQUENCE\_RESETUPs. |
| PF\_Cmd\_SEQUENCE\_FLATTEN | Sent when saving and when duplicating the sequence. Flatten sequence data containing pointers or handles so it can be written to disk. This will saved with the project file. Free the unflat data and set the out\_data>sequence\_data to point to the new flattened data. Flat data must be correctly byte-ordered for file storage.  As of 6.0, if an effect’s sequence data has recently been flattened, the effect may be deleted without receiving an additional PF\_Cmd\_SEQUENCE\_SETDOWN. In this case, After Effects will dispose of your flat sequence data. |
| PF\_Cmd\_SEQUENCE\_SETDOWN | Free all sequence data. |
| Frame Selectors  Passed for each frame (or set of audio samples) to be rendered by your plug-in. | |
| PF\_Cmd\_FRAME\_SETUP | Allocate any frame-specific data. This is sent immediately before each frame is rendered, to allow for frame-specific setup data. If your effect changes the size of its output buffer, specify the new output height, width, and relative origin. All parameters except the input layer are valid.  If you set width and height to 0, After Effects ignores your response to the following *PF\_Cmd\_RENDER*.  NOTE: If [PF\_Outflag\_I\_EXPAND\_BUFFER](#_bookmark152) is set, you will receive this selector (and *PF\_Cmd\_FRAME\_SETDOWN)* twice, once without *PF\_Cmd\_RENDER* between them. This is so we know whether or not the given layer will be visible.  Frame data dates from the days when machines might have 8MB of RAM. Given the calling sequence (above), it’s much more efficient to just allocate during *PF\_Cmd\_RENDER*. |

|  |  |
| --- | --- |
| **Selector** | **Response** |
| PF\_Cmd\_RENDER | Render the effect into the output, based on the input frame and any parameters. This render call can only support 8-bit or 16-bit per channel rendering. 32-bit per channel rendering must be handled in PF\_Cmd\_SMART\_RENDER.  All fields in PF\_InData are valid. If your response to this selector is interrupted (your calls to PF\_ABORT or PF\_PROGRESS returns an error code), your results will not be used. You cannot delete frame\_data during this selector; you must wait until PF\_Cmd\_FRAME\_SETDOWN. |
| PF\_Cmd\_FRAME\_SETDOWN | Free any frame data allocated during PF\_Cmd\_FRAME\_SETUP. |
| PF\_Cmd\_AUDIO\_SETUP | Sent before every audio render. Request a time span of input audio. Allocate and initialize any sequence-specific data. If your effect requires input from a time span other than the output time span, update the startsampL and endsampL field in PF\_OutData. |
| PF\_Cmd\_AUDIO\_RENDER | Populate [PF\_OutData](#_bookmark127).dest\_snd with effect-ed audio. All fields in PF\_InData are valid. If your response to this selector is interrupted (your calls to [PF\_ABORT](#_bookmark284) or [PF\_PROGRESS](#_bookmark286) returns an error code), your results will not be used. |
| PF\_Cmd\_AUDIO\_SETDOWN | Free memory allocated during PF\_Cmd\_AUDIO\_SETUP. |
| PF\_Cmd\_SMART\_PRE\_RENDER | SmartFX only. Identify the area(s) of input the effect will need to produce its output, based on whatever criteria the effect implements. |
| PF\_Cmd\_SMART\_RENDER | SmartFX only. Perform rendering and provide output for the area(s) the effect was asked to render. |
| Messaging  The communication channel between After Effects and your plug-in. | |
| PF\_Cmd\_EVENT | This selector makes use of the extra parameter; the [*type*](#_bookmark420) [*of event*](#_bookmark420) to be handled is indicated by the e\_type field, a member of the structure pointed to by extra. See [*Effect*](#_bookmark425) [*UI & Events*](#_bookmark425). |

|  |  |
| --- | --- |
| **Selector** | **Response** |
| PF\_Cmd\_USER\_CHANGED\_PARAM | The user changed a parameter value. You will receive this command only if you’ve set the [PF\_ParamFlag\_SUPERVISE](#_bookmark218) flag. You modify the parameter to control values, or make one parameter’s value affect others. A parameter can be modified by different actions.  in\_data.current\_time is set to the time of the frame that the user is looking at in the UI (internally, the current time of the comp converted into layer time) while they are changing the param that triggered the PF\_Cmd\_USER\_CHANGED\_PARAM. It’s also the time of a keyframe that is added automatically (if there isn’t one already, and the stopwatch is enabled).  This is usually the same as the value passed for the PF\_Cmd\_RENDER that follows immediately after (unless caps lock is down), but not necessarily – there could be other comp windows open that cause a render at a different time in response to the changed param. |
| PF\_Cmd\_UPDATE\_PARAMS\_UI | The effect controls palette (ECP) needs to be updated. This might occur after opening the ECP or moving to a new time within the composition. You can modify parameter characteristics (enabling or disabling them, for example) by calling [PF\_UpdateParamUI()](#_bookmark320).  Only cosmetic changes may be made in response to this command. Don’t change parameter values while responding to PF\_Cmd\_UPDATE\_PARAMS\_UI; do so during PF\_Cmd\_USER\_CHANGED\_PARAM instead.  This command will only be sent regularly if [PF\_OutFlag\_SEND\_UPDATE\_PARAMS\_UI](#_bookmark162) was set in the PiPL, and during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75).  NOTE: Never check out parameters during this selector. Recursive badness is almost guaranteed to result. |
| PF\_Cmd\_DO\_DIALOG | Display an options dialog. this is sent when the Options button is clicked (or a menu command has been selected). This selector will only be sent if the effect has previously indicated that it has a dialog (by setting the global PF\_OutFlag\_I\_DO\_DIALOG flag in response to PF\_Cmd\_GLOBAL\_SETUP). in version 3.x, the params passed with PF\_Cmd\_DO\_DIALOG were invalid. This is no longer the case; plug-ins can access non- layer parameters, check out parameters at other times, and perform UI updates during PF\_Cmd\_DO\_DIALOG.They still may not change the parameter’s values. |

|  |  |
| --- | --- |
| **Selector** | **Response** |
| PF\_Cmd\_ARBITRARY\_CALLBACK | Manage your arbitrary data type. You’ll only receive this if you’ve registered a custom data type parameter. The extra parameter indicates which handler function is being called. Custom data types are discussed further in [*Implementation*](#_bookmark335). |
| PF\_Cmd\_GET\_EXTERNAL\_DEPENDENCIES | Only sent if  [PF\_OutFlag\_I\_HAVE\_EXTERNAL\_DEPENDENC](#_bookmark161)  [IES](#_bookmark161) was set during [*P F\_Cmd\_GLOBAL\_SETUP*](#_bookmark75). Populate a string handle (in the PF\_ExtDependenciesExtra pointed to by extra) with a description of your plug-in’s dependencies, making sure to allocate space for the terminating NULL character. Return just a NULL pointer for the string handle if there are no dependencies to report.  If the check type is PF\_DepCheckType\_ALL\_DEPENDENCIES, report everything that might be required for your plug-in to render. Report only missing items (or a null string if nothing’s missing) if the check type is PF\_DepCheckType\_MISSING\_DEPENDENCIES. |

|  |  |
| --- | --- |
| **Selector** | **Response** |
| PF\_Cmd\_COMPLETELY\_GENERAL | Respond to an AEGP. The extra parameter points to whatever parameter the AEGP sent. AEGPs can only communicate with effects which respond to this selector. |
| PF\_Cmd\_QUERY\_DYNAMIC\_FLAGS | New in 5.0. Sent only to plug-ins which have specified  PF\_OutFlag2\_SUPPORTS\_QUERY\_DYNAMIC\_F  LAGS in PF\_OutFlags2, in their PiPL and during PF\_Cmd\_GLOBAL\_SETUP. With all of the dynamic flags, if you will ever change them during this command, you must have set the flag on during PF\_Cmd\_GLOBAL\_SETUP.  This selector will be sent at arbitrary times. In response, the effect should access its (non-layer) parameters, and decide whether any of the flags that support PF\_Cmd\_QUERY\_DYNAMIC\_FLAGS should be set, such as:  [PF\_OutFlag\_WIDE\_TIME\_INPUT](#_bookmark145) [PF\_OutFlag\_NON\_PARAM\_VARY](#_bookmark148) [PF\_OutFlag\_PIX\_INDEPENDENT](#_bookmark153) [PF\_OutFlag\_I\_USE\_SHUTTER\_ANGLE](#_bookmark158) [PF\_OutFlag2\_I\_USE\_3D\_CAMERA](#_bookmark169) [PF\_OutFlag2\_I\_USE\_3D\_LIGHTS](#_bookmark171) [PF\_OutFlag2\_DOESNT\_NEED\_EMPTY\_PIXELS](#_bookmark174) [PF\_OutFlag2\_REVEALS\_ZERO\_ALPHA](#_bookmark175) [PF\_OutFlag2\_DEPENDS\_ON\_UNREFERENCED](#_bookmark181)  [\_MASKS](#_bookmark181) [PF\_OutFlag2\_OUTPUT\_IS\_WATERMARKED](#_bookmark183)  After Effects uses this information for caching and optimization purposes, so try to respond as quickly as possible. |

**WHAT’S THE DIFFERENCE?**

There is a subtle difference between [PF\_Cmd\_USER\_CHANGED\_PARAM](#_bookmark101) and

[PF\_Cmd\_UPDATE\_PARAMS\_UI](#_bookmark103). Effects need to distinguish between the user actually

changing a parameter value (PF\_Cmd\_USER\_CHANGED\_PARAM*)*, and just scrubbing around the timeline (PF\_Cmd\_UPDATE\_PARAMS\_UI*,* which is also sent when the plug-in is first

loaded).

**PF\_INDATA**

After Effects communicates system, project, layer and audio information using PF\_InData. This structure is updated before each command selector is sent to a plug-in. Fields valid only during specific [PF\_Cmds](#_bookmark72) are noted. Also, don’t worry; although PF\_InData is dauntingly

large, you need not memorize each member’s purpose; you’ll use some of the fields some of the time.

**TABLE 5: PF\_INDATA**

|  |  |
| --- | --- |
| **Name** | **Description** |
| inter | Callbacks used for user interaction, adding parameters, checking whether the user has interrupted the effect, displaying a progress bar, and obtaining source frames and parameter values at times other than the current time being rendered. This very useful function suite is described in [*I nteraction Callback Functions*](#_bookmark280). |
| utils | Graphical and mathematical callbacks. This pointer is defined at all times. |
| effect\_ref | Opaque data that must be passed to most of the various callback routines. After Effects uses this to identify your plug-in. |
| quality | The current quality setting, either PF\_Quality\_HI or PF\_Quality\_LO. Effects should perform faster in LO, and more accurately in HI. The graphics utility callbacks perform differently between LO and HI quality; so should your effect! This field is defined during all frame and sequence selectors. |
| version | Effects specification version, Indicate the version you need to run successfully during  PF\_Cmd\_GLOBAL\_SETUP. |
| serial\_num | The serial number of the invoking application. |
| appl\_id | The identifier of the invoking application. If your plug-in is running in After Effects, appl\_id contains the application creator code ‘FXTC ’. If it is running in [Premiere](#_bookmark818)  [Pro](#_bookmark818), it will be ‘PrMr’. Use this to test whether your plug-in, licensed for use with one application, is being used with another. |
| num\_params | Input parameter count. |
| what\_cpu | Under Mac OS this contains the Gestalt value for CPU type (see Inside Macintosh, volume 6). Undefined on Windows. |
| what\_fpu | Under Mac OS this contains the Gestalt value for FPU type. Undefined on Windows. |

|  |  |
| --- | --- |
| **Name** | **Description** |
| current\_time | The time of the current frame being rendered, valid during [*PF\_Cmd\_RENDER*](#_bookmark88). This is the current time in the layer, not in any composition. If a layer starts at other than time 0 or is time-stretched, layer time and composition time are distinct.  The current frame number is current\_time divided by time\_step. The current time in seconds is current\_time divided by time\_scale.  To handle time stretching, composition frame rate changes, and time remapping, After Effects may ask effects to render at non-integral times (between two frames). Be prepared for this; don’t assume that you’ll only be asked for frames on frame boundaries. NOTE: As of CS3 (8.0), effects may be asked to render at negative current times. Deal! |
| time\_step | The duration of the current source frame being rendered. In several situations with nested compositions, this source frame duration may be different than the time span between frames in the layer (local\_time\_step). This value can be converted to seconds by dividing by time\_scale.  When calculating other source frame times, such as for [PF\_CHECKOUT\_PARAM](#_bookmark288), use this value rather than local\_time\_step.  Can be negative if the layer is time-reversed. Can vary from one frame to the next if time remapping is applied on a nested composition.  Can differ from local\_time\_step when source material is stretched or remapped in a nested composition. For example, this could occur when an inner composition is nested within an outer composition with a different frame rate, or time remapping is applied to the outer composition.  This value will be 0 during [*PF\_Cmd\_SEQUENCE\_SETUP*](#_bookmark79) if it is not constant for all frames. It will be set correctly during PF\_Cmd\_FRAME\_SETUP and PF\_Cmd\_FRAME\_SETDOWN selectors. WARNING: This can be zero, so check it before you divide. |
| total\_time | Duration of the layer. If the layer is time-stretched longer than 100%, the value will be adjusted accordingly; but if the layer is time-stretched shorter, the value will not be affected. If time remapping is enabled, this value will be the duration of the composition. This value can be converted to seconds by dividing by time\_scale. |
| local\_time\_step | Time difference between frames in the layer. Affected by any time stretch applied to a layer. Can be negative if the layer is time-reversed. Unlike time\_step, this value is constant from one frame to the next. This value can be converted to seconds by dividing by time\_scale.  For a step value that is constant over the entire frame range of the layer, use local\_time\_step, which is based on the composition’s framerate and layer stretch. |

|  |  |
| --- | --- |
| **Name** | **Description** |
| time\_scale | The units per second that current\_time, time\_step, local\_time\_step and total\_time are in. If time\_scale is 30, then the units of current\_time, time\_step, local\_time\_step and total\_time are in 30ths of a second. The time\_step might then be 3,  indicating that the sequence is actually being rendered at 10 frames per second.  total\_time might be 105, indicating that the sequence is 3.5 seconds long. |
| field | Valid only if [PF\_OutFlag\_PIX\_INDEPENDENT](#_bookmark153) was set during  [*PF\_Cmd\_GL OBAL\_SETUP*](#_bookmark75). Check this field to see if you can process just the upper or lower field. |
| shutter\_angle | Motion blur shutter angle. Values range from 0 to 1, which represents 360  degrees. Will be zero unless motion blur is enabled and checked for the target layer. shutter\_angle [== 180 means the time interval between](#_bookmark158)  [current\_time and current\_time + 1/2 time\_step. Valid only if](#_bookmark158) [P](#_bookmark158)F\_OutFlag\_I\_USE\_SHUTTER\_ANGLE was set during [*PF\_Cmd\_GL OBAL\_SETUP*](#_bookmark75).  See the section on [*Motion Blur*](#_bookmark362) for details on how to implement motion blur in your effect. |
| width | Dimensions of the source layer, which are not necessarily the same as the width and height fields in the input image parameter. Buffer resizing effects can cause this difference. Not affected by downsampling. |
| height |
| extent\_hint | The intersection of the visible portions of the input and output layers; encloses the composition rectangle transformed into layer coordinates. Iterating over only this rectangle of pixels can speed your effect dramatically. See [*notes*](#_bookmark124) later in this chapter regarding proper usage. |
| output\_origin\_x | The origin of the output buffer in the input buffer. Non-zero only when the effect changes the origin. |
| output\_origin\_y |
| downsample\_x | [Point control parameters and layer parameter dimensions are automatically adjusted to compensate for a user telling After Effects to render only every nth pixel. Effects need the downsampling factors to interpret scalar parameters representing pixel distances in the image (like sliders). For example, a blur of 4 pixels should be interpreted as a blur of 2 pixels if the downsample factor is 1/2 in each direction (downsample factors are represented as ratios.) Valid only during](#_bookmark79)  [*PF\_Cmd\_*](#_bookmark79)*SEQUENCE\_SETUP*,  [*PF\_Cmd\_SEQUENCE\_RESETUP*](#_bookmark81),  [*PF\_Cmd\_FRAME\_SETUP*](#_bookmark86) and  [*PF\_Cmd\_FRAME\_RENDER*](#_bookmark88). |
| downsample\_y |
| pixel\_aspect\_ ratio | Pixel aspect ratio (width over height). |
| in\_flags | Unused. |

|  |  |
| --- | --- |
| **Name** | **Description** |
| global\_data | Data stored by your plug-in during other selectors. Locked and unlocked by After Effects before and after calling the plug-in. |
| sequence\_data |
| frame\_data |
| start\_sampL | Starting sample number, relative to the start of the audio layer. |
| dur\_sampL | Duration of audio, expressed as the number of samples. Audio-specific. |
| total\_sampL | Samples in the audio layer; equivalent to total\_time expressed in samples. |
| src\_snd | PF\_SoundWorld describing the input sound. Audio-specific. |
| pica\_basicP | Pointer to the PICA Basic suite, used to acquire other suites. |
| pre\_effect\_ source\_origin\_x | Origin of the source image in the input buffer. Valid only when sent with a frame selector. Non-zero only if one or more effects that preceded this effect on the same layer resized the output buffer and moved the origin. Check for both the resize and the new origin to determine output area. This is useful for effects which have implicit spatial operations (other than point controls), like flipping a file around an image’s center.  NOTE: Checked-out point parameters are adjusted for the pre-effect origin at the current time, not the time being checked out. |
| pre\_effect\_ source\_origin\_y |
| shutter\_phase | Offset from frame time to shutter open time as a percentage of a frame duration. |

**EXTENT\_HINT USAGE**

*(Note: hint rectangles are much more effective...and complicated...for* [*SmartFX*](#_bookmark402)*.)*

Use extent\_hint to process only those pixels for which output is required; this is one of the simplest optimizations you can make. Tell After Effects you use in\_data>[extent\_hint](#_bookmark118) by setting [PF\_OutFlag\_USE\_OUTPUT\_EXTENT](#_bookmark150) in [PF\_OutData](#_bookmark127) during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) (and in your PiPL).

Disable caching from the preferences menu before testing extent\_hint code, so After Effects renders your effect whenever anything in your composition changes. Otherwise, the caching mechanism would obscure your plug-in’s (possibly incorrect) output.

Move the layer within the composition so it’s cropped. The output>extent\_hint is the

portion of the layer which is visible in the composition. Add a mask to your layer and move it around. This changes the extent\_hint, which encloses all of the non-zero alpha areas of the image. The in\_data>extent\_hint is the intersection of these two rectangles (the composition and the mask), and changes whenever they do.

Extent rectangles are computed in the coordinate space of the original input layer, before resizing and origin shifting, to simplify rectangle intersection between the input and output extents for effects which set [PF\_OutFlag\_PIX\_INDEPENDENT](#_bookmark153). To get the output extent in the coordinate system of the output buffer, offset the extent\_hint by the [PF\_InData>](#_bookmark120)output\_origin\_x and [y](#_bookmark121) fields.

Account for downsampling when computing output size; users must be able to render at full resolution. If the output buffer exceeds 30,000 by 30,000, clamp it to that size, and consider displaying an alert dialog.

Once your code behaves correctly, enable the cache and see how frequently the effect needs to re-render. Consider a drop shadow; users frequently apply a static drop shadow to a still

image. The output>extent\_hint is ignored, so the cache is used more often.

For buffer-expanding effects, intersect the output>extent\_hint with your plug-in’s transformed bounds and sets the size accordingly during [*PF\_Cmd\_FRAME\_SETUP*](#_bookmark86).

**NOW WITH 20% MORE PIXELS!**

As of 6.0, the extent\_hints passed are 20% larger than the layer itself, to help with our predictive rendering decisions. Numerous effects expand the buffer “just a touch”, and After Effects often uses the hint rectangles later.

**POINT CONTROLS AND BUFFER EXPANSION**

Effects which expand the output buffer position the original layer’s upper left corner by

setting set [output\_origin\_x/y](#_bookmark120) in [PF\_InData](#_bookmark110) during [*PF\_Cmd\_FRAME\_SETUP*](#_bookmark86). This shift is reported to subsequent effects in the [pre\_effect\_source\_origin\_x/y](#_bookmark122). Point parameters are adjusted for this shift automatically.

Apply a buffer expander such as Gaussian Blur or the Resizer SDK sample, *before* your effect, and use a large resize value. If your effect is not handling pre\_effect\_source\_origin\_x/y correctly, turning the blur on and off will shift the position of the output.

All point parameter values (at any time) have shift values described by pre\_effect\_source\_origin\_x/y. For most effects this works transparently. However, if a buffer expansion changes over time (as with an animated blur amount), the origin shift will move non-animated points. Consider this when designing effects which cache point

parameter values between frames.

**PF\_OUTDATA**

Communicate changes made by your plug-in to After Effects using PF\_OutData. Valid times for altering these fields are noted.

**TABLE 6: PF\_OUTDATA**

|  |  |
| --- | --- |
| **Field** | **Description** |
| my\_version | Set this flag (using the PF\_VERSION macro) to the version of your plug-in code. After Effects uses this data to decide which of duplicate effects to load. |
| name | Unused. |
| global\_data | Handle which will be returned to you in [PF\_InData](#_bookmark110) with every call. Use After Effects’ memory allocation functions. |
| num\_params | After Effects checks this field against the number of calls made to  PF\_ADD\_PARAM, as well as the implicit input layer. |
| sequence\_data | Allocatable upon receiving [PF\_Cmd\_SEQUENCE\_SETUP](#_bookmark79), this handle will be passed back to you in [PF\_InData](#_bookmark110) during all subsequent calls. |
| flat\_sdata\_size | Unused (After Effects knows the size, because you used its allocation functions to get the memory in the first place). |
| frame\_data | Handle you (might have) allocated during [PF\_Cmd\_FRAME\_SETUP](#_bookmark86). This is never written to disk; it was used to pass information from your [PF\_Cmd\_FRAME\_SETUP](#_bookmark86) response to your [PF\_Cmd\_RENDER](#_bookmark88) or [PF\_Cmd\_FRAME\_SETDOWN](#_bookmark90) (which you must do if you resize the output buffer). Otherwise, this memory is rarely used. |
| width, height, origin | Set during [PF\_Cmd\_FRAME\_SETUP](#_bookmark86) if the output image size differs from the input. width and height are the size of the output buffer, and origin is the point the input should map to in the output. To create a 5- pixel drop shadow up and left, set origin to (5, 5). |
| [out\_flags](#_bookmark142) | Send messages to After Effects. OR together multiple values. |
| return\_msg | After Effects displays any C string you put here (checked and cleared after every command selector). |
| start\_sampL | Used only for [*audio*](#_bookmark514) commands |
| dur\_sampL |
| dest\_snd |
| [out\_flags2](#_bookmark166) | Send messages to After Effects. OR together multiple values. |

These flags communicate capability and status information to After Effects. In previous

versions they were also used to send rudimentary messages, e.g. refresh the UI, send an error message. These capabilities have been supplanted by function suites, and all new messaging functions will come in that format. However, capability flags are still contained in the [PiPL](#_bookmark49).

Update both the PiPL and your source code when you make a change. Many of these flags can be changed during an After Effects session.

**TABLE 7: PF\_OUTFLAGS**

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag\_KEEP\_RESOURCE\_OPEN | The plug-in’s resources must be available during all commands. During [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75), the plug-in’s resources are always open, but unavailable at all other times (except during [PF\_Cmd\_ABOUT](#_bookmark73) and [PF\_Cmd\_DO\_DIALOG](#_bookmark104)), unless this flag has been set. Set if you need access to resources at any time other than during [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75). NOTE: We recommend the plug-in load and store the necessary resources in global data, rather than keeping the file’s resources open. |
| PF\_OutFlag\_WIDE\_TIME\_INPUT | The effect checks out a parameter at a time other than [current\_time](#_bookmark115). If you use a parameter (including layer parameters) from another time, set this flag. Otherwise, After Effects won’t correctly invalidate cached frames used by your effect. Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75).  If you set this flag, we strongly recommend you also set [PF\_OutFlag2\_AUTOMATIC\_WIDE\_ TIME\_INPUT](#_bookmark179) for better performance. |
| PF\_OutFlag\_NON\_PARAM\_VARY | With this flag set, After Effects will not cache output when the effect is applied to a still. Otherwise, After Effects will cache your output to be used to render other frames, if possible.  Set this flag if output varies based on something besides a parameter value. If the effect produces changing frames when applied to a still image and all parameters are constant, that's a sure sign that this bit should be set (e.g. Wave Warp).  Particle effects, for example, will need this.  Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75). Can be over- ridden dynamically if needed during  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108). Turn this off whenever possible to improve performance. |
| PF\_OutFlag\_RESERVED6 | Unused. Formerly PF\_OutFlag\_SEND\_PARAMS\_UPDATE. Replaced by [*PF\_OutFlag\_REFRESH\_UI*](#_bookmark156). |

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag\_SEQUENCE\_DATA\_ NEEDS\_FLATTENING | Both After Effects and Premiere Pro assume this flag is set. Flattening is necessary when sequence data contains referencing items (pointers, handles), which must be flattened for storage and unflattened for use. See  [*PF\_Cmd\_ SEQUENCE\_RESETUP*](#_bookmark81). |
| PF\_OutFlag\_I\_DO\_DIALOG | Effect displays a dialog in response to  [*PF\_Cmd\_ DO\_DIALOG*](#_bookmark104). Set during  [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75), checked during  [*PF\_Cmd\_ SEQUENCE\_SETUP*](#_bookmark79).  Note: the effect’s response to PF\_OutFlag\_I\_DO\_DIALOG is not undoable. You can use arbitrary data with a custom UI, should such changes become necessary. |
| PF\_OutFlag\_USE\_OUTPUT\_EXTENT | Effect honors the output [extent\_rect](#_bookmark118). Set during  [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75). See details at the end of the chapter for proper usage.  Note: Obsolete for SmartFX. |
| PF\_OutFlag\_SEND\_DO\_DIALOG | Effect must show dialog to function (added for compatibility with Photoshop plug-ins). After Effects sends  [*PF\_Cmd\_ DO\_DIALOG*](#_bookmark104) after  [*PF\_Cmd\_ SEQUENCE\_SETUP*](#_bookmark79). Set during  [*PF\_Cmd\_ SEQUENCE\_RESETUP*](#_bookmark81), not during  [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75). |
| PF\_OutFlag\_DISPLAY\_ERROR\_MESSAGE | Display the contents of [return\_msg](#_bookmark140) in an error dialog. Whenever return\_msg is non-NULL, After Effects displays the contents in a dialog, which will be an error dialog if this flag is set. Set after any command, and can be used during debugging. This is also a good way to implement nag messages for tryout versions. |
| PF\_OutFlag\_I\_EXPAND\_BUFFER | Effect expands the output buffer. Set during  [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75). Set this flag and [PF\_OutFlag\_USE\_OUTPUT\_EXTENT](#_bookmark150) to use the intersection of the output [extent\_rect](#_bookmark118) and your new buffer size during [*PF\_Cmd\_FRAME\_SETUP*](#_bookmark86). Use [pre\_effect\_source\_origin](#_bookmark122) fields to detect other transformations.  Note: Only set this flag if you need to; it drastically reduces caching efficiency.  Note: Obsolete for SmartFX. |

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag\_PIX\_INDEPENDENT | A given pixel is independent of the pixels around it. Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) or  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108). As an example, color correction effects are typically pixel independent, distortions are not.  NOTE: If your effect doesn’t use the color values of one pixel to affect those of adjacent pixels, set this outflag! It can provide dramatic performance improvements. |
| PF\_OutFlag\_I\_WRITE\_INPUT\_BUFFER | The effect writes into the input buffer. This is of limited use; while saving an allocation, it invalidates some pipeline caching. Set during [*PF\_Cm d\_GLOBAL\_SETUP*](#_bookmark75). |
| PF\_OutFlag\_I\_SHRINK\_BUFFER | The effect shrinks its buffer based on the [extent\_rect](#_bookmark118)  in order to be more memory efficient. Set during  [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75) whenever possible. Note: Obsolete for SmartFX. |
| PF\_OutFlag\_WORKS\_IN\_PLACE | Unused. |
| PF\_OutFlag\_SQUARE\_PIX\_ONLY | Unused. |
| PF\_OutFlag\_CUSTOM\_UI | The effect has a custom user interface and requires  [*PF\_Cmd\_ EVENT*](#_bookmark99) messages. Set during  [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75). |
| PF\_OutFlag\_RESERVED5 | Unused. |
| PF\_OutFlag\_REFRESH\_UI | Refresh the entire effect controls, composition, and layer windows. Set during [*PF\_Cmd\_ EVENT*](#_bookmark99),  [*PF\_Cmd\_ RENDER*](#_bookmark88), and [*PF\_Cmd\_DO\_DIALOG*](#_bookmark104). If refreshing custom UI during PF\_Cmd\_EVENT, we recommend using the [*new redraw mechanism*](#_bookmark480) with finer granularity. |
| PF\_OutFlag\_NOP\_RENDER | Set this flag during [*P F\_Cmd\_FRAME\_SETUP*](#_bookmark86) to invalidate the current render. |
| PF\_OutFlag\_I\_USE\_SHUTTER\_ANGLE | Indicates rendered images depend upon the value of  [shutter\_angle](#_bookmark116). |
| PF\_OutFlag\_I\_USE\_AUDIO | Effect’s parameters depend on audio data, obtained using  [PF\_CHECKOUT\_LAYER\_AUDIO](#_bookmark294). |
| PF\_OutFlag\_I\_AM\_OBSOLETE | Effect is available for use when working with an old project in which it was originally applied, but doesn’t appear in the effect menu. |

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag\_FORCE\_RERENDER | Effect made a change that requires a re-render. PF\_ChangeFlag\_CHANGED\_VALUE also forces a re-render. |
| PF\_OutFlag\_PiPL\_OVERRIDES  \_OUTDATA\_OUTFLAGS | After Effects will use PiPL outflags, and ignore those set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75). |
| PF\_OutFlag\_I\_HAVE\_EXTERNAL\_ DEPENDENCIES | Effect depends on an external file (or external font). If set, After Effects sends  [*PF\_Cmd\_ GET\_EXTERNAL\_DEPENDENCIES*](#_bookmark106). |
| PF\_OutFlag\_DEEP\_COLOR\_AWARE | The effect handles 16-bpc color. |
| PF\_OutFlag\_SEND\_UPDATE\_PARAMS\_UI | Set this flag during [*P F\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) to receive [*P F\_Cmd\_UPDATE\_PARAMS\_UI*](#_bookmark103). |
| PF\_OutFlag\_AUDIO\_FLOAT\_ONLY | Effect requires audio data in PF\_SIGNED\_FLOAT format. After Effects will perform any required format conversion. You must also set either [PF\_OutFlag\_AUDIO\_EFFECT\_TOO](#_bookmark164) or [PF\_OutFlag\_AUDIO\_EFFECT\_ONLY](#_bookmark165). |
| PF\_OutFlag\_AUDIO\_IIR | Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) if the (audio) effect is an Infinite Impulse Response filter. This is true if output at a given time depends on output from previous times. When an IIR filter receives  [*PF\_Cmd\_ AUDIO\_RENDER*](#_bookmark94), the input audio time span is the same as the output audio time span (when they intersect with the output time span requested in  [*PF\_Cmd\_ AUDIO\_SETUP*](#_bookmark92)). In response to  [*PF\_Cmd\_ AUDIO\_SETUP*](#_bookmark92), the filter can request audio from earlier times (as for delay effects). The filter can access parameters from that earlier time, and should cache them (along with interim audio) in sequence data. If the audio generated does not correspond to the requested output audio’s time, the output audio duration should be set to zero. The filter can update its delay line using the parameters and the input audio. Having cached its delay line, request more input audio during [*PF\_Cmd\_AUDIO\_SETUP*](#_bookmark92) based on the last cached delay line. Use [PF\_HasParamChanged](#_bookmark326) to determine whether or not your cache is valid. |
| PF\_OutFlag\_I\_SYNTHESIZE\_AUDIO | Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) time if the effect generates audio, even when passed silence. You must also set either [PF\_OutFlag\_AUDIO\_EFFECT\_TOO](#_bookmark164) or [PF\_OutFlag\_AUDIO\_EFFECT\_ONLY](#_bookmark165). |
| PF\_OutFlag\_AUDIO\_EFFECT\_TOO | Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) if the effect alters audio. |

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag\_AUDIO\_EFFECT\_ONLY | Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) if the effect alters only audio output. |

**PF\_OUTFLAGS2**

We added a second set of outflags in After Effects 5.0; partly for room to expand in the

future, and partly to break ourselves of the bad habit of repurposing existing flags. As with PF\_OutFlags, many of these flags can be changed during an After Effects session. And don’t forget to update both the [PiPL](#_bookmark49) and your source code when you make a change.

**TABLE 8: PF\_OUTFLAGS2**

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag2\_NONE | Nothing. |
| PF\_OutFlag2\_SUPPORTS\_QUERY\_ DYNAMIC\_FLAGS | The effect responds to  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108). Must be set in the PiPL and during [*P F\_Cmd\_GLOBAL\_SETUP*](#_bookmark75). |
| PF\_OutFlag2\_I\_USE\_3D\_CAMERA | The effect accesses 3D camera information. |
| PF\_OutFlag2\_I\_USE\_3D\_LIGHTS | The effect accesses 3D lighting information. |
| PF\_OutFlag2\_PARAM\_GROUP\_START\_ COLLAPSED\_FLAG | This flag in itself doesn’t control the state of the param group twirlies. The initial collapse state of each individual parameter group is set during [*P*](#_bookmark77)*F\_Cmd\_PARAM\_SETUP*, by setting the [PF\_ParamFlag\_START\_COLLAPSED](#_bookmark220) flag in [PF\_ParamFlags](#_bookmark216). But those individual settings will not be honored unless the effect sets this bit. Otherwise, all parameter groups will be collapsed by default. Remember to set this flag in both the PiPL and here during  [*PF\_Cmd\_ GLOBAL\_SETUP.*](#_bookmark75) |
| PF\_OutFlag2\_I\_AM\_THREADSAFE | Currently this does nothing. If this sounds interesting to you, you may be interested in PF\_OutFlag2\_PPRO\_DO\_NOT\_CLONE\_ SEQUENCE\_DATA\_FOR\_RENDER, described below. |
| PF\_OutFlag2\_CAN\_COMBINE\_WITH\_ DESTINATION | Originally added for Premiere usage, but no longer used. |

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag2\_DOESNT\_NEED\_ EMPTY\_PIXELS | Added for render optimizations; shrinks the input buffer passed to the effect to exclude any empty pixels (where empty means "zero alpha" unless PF\_OutFlag2\_REVEALS\_ZERO\_ALPHA is set, in which case RGB must be zero as well).  Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) or  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108). The origin of the trimmed buffer can be found in in\_data>[pre\_effect\_source\_origin](#_bookmark122). Effects with both this flag and PF\_OutFlag\_I\_EXPAND\_BUFFER set may get called with a null input buffer if their input is completely empty, and must be able to handle this case without crashing.  Note: this flag can cause the size of the output buffer to change. Note: Obsolete for SmartFX. |
| PF\_OutFlag2\_REVEALS\_ZERO\_ALPHA | This is the one flag implementors need to pay most attention to since it represents a change in the default behavior. Set this flag if the effect can take pixels with zero alpha and reveal the RGB data in them (like our Set Channels effect). This tells After Effects not to trim such pixels when determining the input for the effect. This flag can be changed during  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108). Note that, while this flag can cause changes to the size of the [extent\_hint](#_bookmark236), it will not change the image buffer size.  As of 6.0, pixels outside the mask’s bounding box are zeroed. If your effect can reveal such pixels, tell AE not to throw away these RGB values by setting this flag. If your effect does not always reveal such pixels, set this bit dynamically.  To see if your effect needs this bit set, apply a mask significantly smaller than the layer to a solid, then apply the effect and set it to its alpha-modifying state. If the rectangular bounding box of the mask becomes visible, this bit needs to be set. |
| PF\_OutFlag2\_PRESERVES\_FULLY  \_OPAQUE\_PIXELS | Preserve those pixels! |
| PF\_OutFlag2\_SUPPORTS\_ SMART\_RENDER | The effect uses the SmartFX API. |
| PF\_OutFlag2\_FLOAT\_COLOR\_AWARE | The effect supports 32-bpc floating point color representation.  NOTE: PF\_OutFlag2\_SUPPORTS\_ SMART\_RENDER must also be set. |

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag2\_I\_USE\_COLORSPACE\_ ENUMERATION | This is for effects which optimized for different color spaces in Premiere Pro. See the Premiere Pro SDK for more details. |
| PF\_OutFlag2\_I\_AM\_DEPRECATED | Setting this during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) puts the effect in the localized “Obsolete” folder in the Effects panel. Compare to [PF\_OutFlag\_I\_AM\_OBSOLETE](#_bookmark160). |
| PF\_OutFlag2\_PPRO\_DO\_NOT\_CLONE\_ SEQUENCE\_DATA\_FOR\_RENDER | Supported in Premiere Pro, and not in After Effects. This affects how Premiere Pro drives the plug-in using  [*multithreading*](#_bookmark829). |
| PF\_OutFlag2\_AUTOMATIC\_WIDE\_- TIME\_INPUT | Set during [*PF\_Cmd\_GL OBAL\_SETUP*](#_bookmark75). Requires setting of [*PF\_OutFlag\_WIDE\_TIME\_INPUT*](#_bookmark145) (which allows you to support old hosts), but effectively overrides that flag.  When set, all parameter checkouts are tracked so over-time dependencies are known by the host, and much more efficient. For example, if you set only the old  PF\_OutFlag\_WIDE\_TIME\_INPUT, anytime anything changes at any time upstream from your effect, you will be called to re-render. With this flag set, if a given frame 17 has checked out things from times 0-17, AE will know that any changes at frames 18+ will not affect that cached frame.  Note that if you use this new flag, you must not cache any time-dependent data in your sequence data (or anywhere else), unless you also [*validate that cache*](#_bookmark331) using [PF\_GetCurrentState()](#_bookmark322) / [PF\_AreStatesIdentical()](#_bookmark324) before using the time- dependent data.  This only works for SmartFX (those that set  PF\_OutFlag2\_SUPPORTS\_SMART\_RENDER). If  you haven't set that, After Effects will silently treat this as  PF\_OutFlag\_WIDE\_TIME\_INPUT instead. |
| PF\_OutFlag2\_I\_USE\_COMP\_TIMECODE | Set during [*PF\_Cmd\_GLOBAL\_SETUP*](#_bookmark75). This lets AE know it should rerender an effect if the composition start time and/ or drop-frame setting has been modified. |

|  |  |
| --- | --- |
| **Flag** | **Indicates** |
| PF\_OutFlag2\_DEPENDS\_ON  \_UNREFERENCED\_MASKS | New in CS6. Set this if you are going to look at paths that aren't directly referenced by a path param, e.g. if you are going to draw a stroke on all masks. This is needed so After Effects knows to invalidate your output when a mask is modified that doesn't appear to be referenced by your effect. Set during  [*PF\_Cmd\_ GLOBAL\_SETUP*](#_bookmark75) or  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108). |
| PF\_OutFlag2\_OUTPUT\_IS\_ WATERMARKED | New in CS6. Set this during [*P F\_Cmd\_GLOBAL\_SETUP*](#_bookmark75) if your output is going to be watermarked in some way that makes it unsuitable for final use, probably because the user is using an unlicensed demo version. It is ok to change this state during the course of app session during  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108), if e.g. a floating license status changes.  Plug-in authors that actually do have this state changing asynchronously must be careful to have the next render match the last state returned from  [*PF\_Cmd\_ QUERY\_DYNAMIC\_FLAGS*](#_bookmark108) otherwise race conditions could cause incorrect frames to be cached. (This is a non-issue if you only change this in response to DO\_DIALOG.) |

**PARAMETERS**

Parameters are streams of values that vary with time; the source image, sliders, angles,

points, colors, paths, and any arbitrary data types the user can manipulate. They are passed to the plug-in as an array of PF\_ParamDefs, though the values in the array are only valid

during certain selectors.

One of the best aspects of the After Effects effect API is the parameter interpolation and management. How much does the shutter angle change during one-fourth of a 29.97 fps frame? Not your problem; leave it to After Effects.

Describe your plug-in’s parameters during [*PF\_Cmd\_PARAMS\_SETUP*](#_bookmark77), using [PF\_ADD\_PARAM()](#_bookmark283). You may have up to (approximately) 38 kajillion parameters, or as many as your users are willing to sift through before demanding a refund. Choose wisely.

Avoid countless problems by clearing PF\_ParamDefs with AEFX\_CLR\_STRUCT (defined in

AE\_Macros.h) before registering them.

**TABLE 9: PARAMETER TYPES**

|  |  |  |
| --- | --- | --- |
| **Parameter Type** | **Parameter Type, PF\_ParamDefUnion Member,**  **Param Value Data Type** | **Description** |
| PF\_Param\_LAYER | [PF\_LayerDef](#_bookmark226) ld  A\_long | Image and audio layers in the composition. All effects automatically have at least 1 layer  parameter, param[0], the layer to which they are applied.  When used as effect parameters, these appear as a pull-down menu with which the user  selects a layer within the current composition. The pull-down menu contents are generated by After Effects.  NOTE: This is a reference to a layer which contains pixels and audio samples, not actual pixels and audio samples. |
| PF\_Param\_SLIDER | PF\_SliderDef sd  long | No longer used. |
| PF\_Param\_FIX\_SLIDER | PF\_FixedSliderDef fd  PF\_Fixed | Deprecated. For many years, we promoted fixed sliders. We now recommend  PF\_Param\_FLOAT\_SLIDERs. The additional precision helps in many situations, and isn’t as expensive as it once was. Plus, we’re just tired of low byte / high byte silliness.  FIX\_SLIDERs provide higher precision than PF\_Param\_SLIDER. Specify the UI decimal places independently. Ignore the low word of  the PF\_Fixed to get integral results. |
| PF\_Param\_FLOAT\_- SLIDER | PF\_FloatSliderDef fs\_d  PF\_FPLong | Sliders represent numerical values. FLOAT\_SLIDERs contain values for phase, precision, and curve tolerance for use by audio  filters. Specify a minimum and maximum value, and the user can move a slider or types a number to specify the setting.  PF\_Param\_FLOAT\_SLIDERs also respond to slider flags discussed in [*Audio*](#_bookmark514) *Filters*. |

|  |  |  |
| --- | --- | --- |
| **Parameter Type** | **Parameter Type, PF\_ParamDefUnion Member,**  **Param Value Data Type** | **Description** |
| PF\_Param\_ANGLE | PF\_AngleDef ad  PF\_Fixed | Angles in (fixed point) degrees, accurate to  small fractions of a degree. Users can specify  multiple revolutions, resulting in values greater than 360. |
| PF\_Param\_CHECKBOX | PF\_CheckBoxDef bd  PF\_Boolean | PF\_ParamFlag\_CANNOT\_INTERP is forced on for all checkboxes. |
| PF\_Param\_COLOR | PF\_ColorDef cd  PF\_Pixel | RGB value (alpha is not used) that the user can choose either with the standard color picker or with an eye dropper tool. For floating point accuracy, use the [*PF Color Param Suite*](#_bookmark307) to  retrieve the values. |
| PF\_Param\_POINT | PF\_PointDef td  PF\_Fixed | A two-dimensional point. The point provides x and y values in destination layer space. New in CS5.5, for floating point accuracy, use the [*PF*](#_bookmark309) [*Point Param Suite*](#_bookmark309) to retrieve the values.  Dusty history lesson to follow: Prior to API specification version 12.1 (After Effects 4.0), the default value for the point was between 0 and 100 in fixed point with the radix point at bit 16 (i.e. standard fixed point). Specifying  (50,50) in fixed point yields the center of the image. The value you are returned for a point  control is in absolute pixels with some number of bits of fixed point accuracy. Thus, if you gave (50,50) as the default position and the user  applied the effect to a 640 by 480 layer, the  default value you would be sent would be (320,  240) in Fixed point. Plug-ins which specify API versions before 12.1 will still get the old behavior. |
| PF\_Param\_POPUP | PF\_PopupDef pd  A\_long | List of choices. Build a string in namesptr containing a list of (read-only) pop-up entries (“Entry1|Entry2|Entry3”). After Effects copies the data and creates a pop-up menu. These entries cannot be modified once the parameter is added.  An entry of “(-” will result in a separator  being drawn between previous and subsequent entries. |

|  |  |  |
| --- | --- | --- |
| **Parameter Type** | **Parameter Type, PF\_ParamDefUnion Member,**  **Param Value Data Type** | **Description** |
| PF\_Param\_ARBITRARY\_ DATA | PF\_ArbitraryDef arb\_d  ??? | Custom data type. An [*arbitrary parameter*](#_bookmark335)contains an ID (you can use more than one custom data type in a given effect), a default  value (so After Effects knows what your data type should start as), and a handle to your  actual parameter.  In AE, must specify either PF\_PUI\_TOPIC/ PF\_PUI\_CONTROL or PF\_PUI\_NO\_ECW.  In PPro 8.0 and later, it's okay to set none of those flags, which allows you to see the  parameter's keyframe track on the right side of Effect Controls without creating a custom  control. |
| PF\_Param\_PATH | PF\_PathDef path\_d PF\_PathID | Path parameters are references to masks applied to the same layer as the effect. Path parameter data cannot be accessed directly; use [PF\_PathQuerySuite](#_bookmark372) and PF\_PathDataSuite to manage and  inquire about paths.  PF\_PathDef.path\_id contains the index of the mask selected by the user. A  corresponding AEGP\_MaskRefH can be obtained using [AEGP\_GetLayerMaskByIndex](#_bookmark654). |
| PF\_Param\_GROUP\_START | (none) | Parameter groups (topics) organize parameters into sets. Each group receives its own twirly  and will be indented in the ECP relative to the neighboring parameters or groups. One group can be nested within another. Each twirly can be spun open or closed by the user, or  programatically by the effect. The effect may choose to have certain groups initialized with the twirly spun open, and others with the  twirly spun closed. |
| PF\_Param\_GROUP\_END | (none) |

|  |  |  |
| --- | --- | --- |
| **Parameter Type** | **Parameter Type, PF\_ParamDefUnion Member,**  **Param Value Data Type** | **Description** |
| PF\_Param\_BUTTON | PF\_Button button\_d  (no value) | New in CS5.5 to After Effects. A simple push button. Use [*parameter supervision*](#_bookmark314) to detect when the button is pressed. |
| PF\_Param\_POINT\_3D | PF\_Point3D point3d\_d  PF\_FpLong (3) | New in CS5.5. Unsupported in Premiere Pro. A three-dimensional point. |

**SLIDER RANGE ISSUES?**

If your slider seems disabled but not grayed out, check the valid\_min, slider\_min, valid\_max and slider\_max fields. Is the param a [PF\_Param\_FIX\_SLIDER](#_bookmark191)? If so, did you convert your mins and maxs to reasonable fixed values? If you’re using the macros

provided in AE\_Macros.h, they’re expecting to receive ints; passing fixed point values won’t work.

**POINT PARAMETER ORIGIN**

After Effects modifies any point parameter to account for origin offset, introduced by

“upstream” effects that modify the output dimensions. Even if the ECP UI indicates the value of the point parameter is (0,0), the offset has already been factored in.

**PF\_PARAMDEF**

After Effects passes effects an array of PF\_ParamDefs with each selector, describing the

plug-in’s parameters at the current time. The values in the params array are only valid during some selectors (this is noted in the [selector descriptions](#_bookmark72)).

**PARAM ZERO**

The first parameter, params[0], is the input image (a [PF\_EffectWorld](#_bookmark226)) to which the effect should be applied.

**THE REST OF THE PARAMETERS**

All parameter types are represented by a PF\_ParamDef. Unions are used, so that only the pertinent parts of the PF\_ParamDef need be (or should be) populated.

**TABLE 10: PF\_PARAMDEF**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Name** | **Description** |
| A\_long | id | The ID of this parameter. You can re-order parameters in future versions of your plug-in and not cause users to re- apply your effect, if you maintain the parameter’s ID  across versions. |
| PF\_ChangeFlags | change\_flags | Set if you’ve changed a parameter value. Only valid during drag (not click!) events, [PF\_Cmd\_USER\_CHANGED\_PARAM](#_bookmark101) or [PF\_Cmd\_UPDATE\_PARAMS\_UI](#_bookmark103). |
| [PF\_ParamUIFlags](#_bookmark211) | ui\_flags | Specify a parameter’s UI behavior before adding; only  PF\_PUI\_DISABLED may be set during event handling. |
| A\_short | ui\_width | Width and height of the parameter’s user interface (for non-standard parameters only). |
| A\_short | ui\_height |
| [PF\_ParamType](#_bookmark188) | param\_type | Type of parameter. |
| A\_char[32] | name | Name of parameter. Can be changed during event handling. Yes, longer parameter names have been  requested since After Effects 1.0. Think of adequately describing your world-altering effect in 31 mere  characters as a language challenge, like haiku. |
| [PF\_ParamFlags](#_bookmark216) | flags | Specify a parameter’s UI behavior before adding; only PF\_ParamFlag\_COLLAPSE\_TWIRLY may be set during event handling. |
| PF\_ParamDefUnion | u | A union of all [*possible parameter types*](#_bookmark188). Only the type  specified by param\_type contains meaningful data. |

**PARAMETER UI FLAGS**

Control a parameter’s user interface with these flags. Don’t confuse UI flags with behavior flags; they reside in different fields within your parameter’s definition, and will cause

unpredictable behavior if misapplied.

**TABLE 11: PARAMETER UI FLAGS**

|  |  |
| --- | --- |
| **Flag** | **Description** |
| PF\_PUI\_TOPIC | Set this flag if you handle PF\_Cmd\_EVENTs for the "topic" of the parameter. The "topic" is the portion of the param UI in the Effect Controls Window (ECW) that is still visible when the  twirly-arrow is twirled up for that param.  If you set this flag, you must also set PF\_OutFlag\_CUSTOM\_UI at PF\_Cmd\_GLOBAL\_SETUP time. |
| PF\_PUI\_CONTROL | Set this flag if you handle PF\_Cmd\_EVENTs for the control area (area that becomes invisible when you twirl up a  parameter's spinner) in the ECP. If you set this flag, you must also set PF\_OutFlag\_CUSTOM\_UI at PF\_Cmd\_GLOBAL\_SETUP time. See [*Events*](#_bookmark425) for more details. |
| PF\_PUI\_STD\_CONTROL\_ONLY | Set this flag if you want the standard control only -- No data stream will be associated with this parameter, and thus no keyframes will be available in the Timeline panel.  You might want to do this to control something in your  sequence data with a standard control. Or in your arb data, or custom UI in the comp window, or to group-set multiple other controls.  This flag cannot be used with PF\_Param\_CUSTOM, PF\_Param\_NO\_DATA, PF\_Param\_LAYER, PF\_Param\_ARBITRARY\_DATA, PF\_Param\_PATH.  If you set this flag, you must also set [PF\_ParamFlag\_SUPERVISE](#_bookmark218) (otherwise you would never find out about value changes, and the setting would never be used for anything). This flag does not require that the [PF\_OutFlag\_CUSTOM\_UI](#_bookmark154) flag be set.  If you want a standard control for  PF\_Param\_ARBITRARY\_DATA, just add one (or more) using PF\_PUI\_STD\_CONTROL\_ONLY with the supported param types, and then when handling  [*PF\_Cmd\_ USER\_CHANGED\_PARAM*](#_bookmark101) you can modify your arb data. |

**TABLE 11: PARAMETER UI FLAGS**

|  |  |
| --- | --- |
| **Flag** | **Description** |
| PF\_PUI\_NO\_ECW\_UI | Set this flag if you want no UI to appear in the Effect Controls Window. Presumably, you are setting the value of the parameter through some other method (e.g. custom UI in the comp  window, or while handling  PF\_Cmd\_USER\_CHANGED\_PARAM for a different param  with PF\_ParamFlag\_SUPERVISE set). In AE, this doesn't affect keyframe visibility in the timeline. In PPro it does remove the entire row, so you won't see keyframes. |
| PF\_PUI\_ECW\_SEPARATOR | Not used in After Effects, but used in Premiere. Set this flag if you'd like a thick line above this parameter in the effect control window. This is provided so that parameters can be grouped visually, if needed (without adding groups). This flag can be  changed at runtime through the PF\_UpdateParamUI() method. |
| PF\_PUI\_DISABLED | Disables (grays out) the parameter, usually in response to  [*PF\_Cmd\_ USER\_CHANGED\_PARAM*](#_bookmark101). |
| PF\_PUI\_DONT\_ERASE\_TOPIC | After Effects won’t erase parameter’s topic. |
| PF\_PUI\_DONT\_ERASE\_CONTROL | After Effects won’t erase parameter’s control. |
| PF\_PUI\_RADIO\_BUTTON | Not used in After Effects, but used in Premiere. Display parameter as a radio-button group. Only valid for PF\_Param\_POPUP. |
| PF\_PUI\_INVISIBLE | First supported in Premiere, and now supported in After Effects CS6 and later. This hides the parameter UI in both the Effect  Controls and Timeline.  Premiere only: The flag is dynamic and parameter visibility can be toggled during the [PF\_UpdateParamUI](#_bookmark320) callback. |

In addition to these flags, an effect parameter may be hidden or shown by using

[AEGP\_GetDynamicStreamFlags](#_bookmark636).

**PARAMETER FLAGS**

Behavior flags and UI flags describe different qualities of a parameter. Set them *before* adding the parameter during [*PF\_Cmd\_PARAM\_SETUP*](#_bookmark77). Flags which may be set during events are noted.

**TABLE 12: PARAMETER FLAGS**

|  |  |
| --- | --- |
| **Flag** | **Meaning** |
| PF\_ParamFlag\_CANNOT\_TIME\_VARY | Parameter does not vary with time; no keyframe control will be provided in the Timeline panel. |
| PF\_ParamFlag\_CANNOT\_INTERP | Values are not algebraically interpolated. You can still use discontinuous (hold) interpolation. Useful for parameters which are either on or off. Accelerates rendering. |
| PF\_ParamFlag\_COLLAPSE\_TWIRLY | Set this flag during  [*PF\_Cmd\_ USER\_CHANGED\_PARAM*](#_bookmark101). This bit can now be set & cleared when handling  [*PF\_Cmd\_ UP DATE\_PARAMS\_UI*](#_bookmark103) and  [*PF\_Cmd\_ USER\_CHANGED\_PARAM*](#_bookmark101) messages, so as to twirl your parameters and groups up and down at will. |
| PF\_ParamFlag\_SUPERVISE | Set to receive [*PF\_Cmd\_USER\_CHANGED\_PARAM*](#_bookmark101)  messages for this parameter. See [*P arameter Supervision*](#_bookmark314) for more information. |
| PF\_ParamFlag\_START\_COLLAPSED | Controls the twirl-state of a topic spinner. Can be changed during parameter supervision, not just during  [*PF\_Cmd\_ PARAM\_ SETUP*](#_bookmark77). This flag will not be honored unless [PF\_OutFlag2\_PARAM\_GROUP\_START\_COLLAPSED](#_bookmark173) is  set. |
| PF\_ParamFlag\_USE\_VALUE\_FOR\_OLD  \_PROJECTS | This only affects the loading of projects saved with an  older version of the effect which lacks parameters added later. When set, the PF\_ParamDef.value field set in PF\_ADD\_PARAM() will be used to initialize the missing parameter, but the dephault field will still be used for initial value of the parameter when the effect is newly applied or reset. This is useful for when you want a  parameter to default to one value but need it set to  something else to preserve rendering behavior for older projects. |
| PF\_ParamFlag\_LAYER\_PARAM\_IS\_TRACKMATTE | Premiere Pro only: Only valid for layer parameters.  Indicates that a layer param is used as a track-matte with applied filters. Ignored in After Effects. |

**TABLE 12: PARAMETER FLAGS**

|  |  |
| --- | --- |
| **Flag** | **Meaning** |
| PF\_ParamFlag\_EXCLUDE\_FROM\_HAVE\_ INPUTS\_CHANGED | Only relevant if the effect sets  [PF\_OutFlag2\_AUTOMATIC\_WIDE\_TIME\_INPUT](#_bookmark179)  and will call [PF\_AreStatesIdentical](#_bookmark324) or [PF\_HaveInputsChangedOverTimeSpan](#_bookmark327) |
| PF\_ParamFlag\_SKIP\_REVEAL\_WHEN\_ UNHIDDEN | New in CS6. If this parameter is unhidden, then this flag tells After Effects to not twirl open any parents and to not scroll the parameter into view in the Effect Controls panel and the Timeline panel.  After Effects uses this behavior internally when paint strokes are made, so as not to distract the user by revealing the parameter. However, in another case, when turning on Time Remapping, that parameter is revealed. So we  provide you the same control over parameters in your own effects. |

**PF\_VALUEDISPLAYFLAGS**

Within PF\_ParamDefUnion, PF\_FloatSliderDef and PF\_FixedSliderDef both have a member variable, PF\_ValueDisplayFlags, which allows them to respond to the user’s

pixel value display preference (which they set in the info palette). If this is set, the

parameter’s value will be displayed as 0-1, 0-255, 0-32768, or 0.0 to 1.0, depending on the preference. You can also set the first bit (PF\_ValueDisplayFlag\_PERCENT) to append a percent sign to the parameter’s displayed value.

We know you’d never do anything like this, but if you create a parameter which displays as a percentage, don’t confuse the user by allowing any range other than 0 to 100. Please. Percent means ‘out of one hundred’.

**PF\_EFFECTWORLD /** **PF\_LAYERDEF**

After Effects represents images using PF\_EffectWorlds, also called PF\_LayerDefs.

**TABLE 13: PF\_EFFECTWORLD STRUCTURE**

|  |  |
| --- | --- |
| **Item** | **Description** |
| world\_flags | Currently, the only flags are:  PF\_WorldFlag\_DEEP - set if the world is 16-bpc  PF\_WorldFlag\_WRITEABLE - indicates that you are allowed to alter the image data of the world. Normally effects cannot alter input image data; only output. |
| data | Pointer to image data, stored as a PF\_PixelPtr. Do not access directly; use the [*accessor macros*](#_bookmark242).  Image data in After Effects is always organized in sequential words each containing Alpha, Red, Green, Blue from the low byte to the high byte. |
| rowbytes | The length, in bytes, of each row in the image’s block of pixels. The block of pixels contains height lines each with width pixels  followed by some bytes of padding. The width pixels (times four,  because each pixel is four bytes long) plus optional extra padding adds up to rowbytes bytes. Use this value to traverse the image data.  Platform-specific padding at the end of rows makes it unwise to  traverse the entire buffer. Instead, find the beginning of each row using  height and rowbytes.  NOTE: This value does not vary based on whether field rendering is active.  NOTE: Input and output worlds with the same dimensions can use different rowbytes values. |
| width | Width and height of the pixel buffer. |
| height |
| extent\_hint | The smallest rectangle encompassing all opaque (non-zero alpha) pixels in the layer. This defines the area which needs to be output. If your plug-in varies with extent (like a diffusion dither), ignore this and render the full frame each time. |

**TABLE 13: PF\_EFFECTWORLD STRUCTURE**

|  |  |
| --- | --- |
| **Item** | **Description** |
| pix\_aspect\_ratio | The pixel aspect ratio expressed as a PF\_Rational. NOTE: Effects can use this value for checked out layers, but must use PF\_InData.pixel\_aspect\_ratio for the layer to which  they’re applied. Sorry. |
| platform\_ref | No longer used in CS5.  Platform-specific reference information. On Windows, this contains an opaque value. Under Mac OS, PF\_GET\_PLATFORM\_REFS provides a CGrafPtr and a GDeviceHandle from a PF\_EffectWorld.  NOTE: You cannot acquire a platform\_ref during  *PF\_Cmd\_GLOBAL\_SETUP*, as there isn’t any output context yet. Patience, my pet. |

**ROWBYTES IN PF\_EFFECTWORLDS**

Don't assume that you can get to the next scanline of a [PF\_EffectWorld](#_bookmark226) using (width \* sizeof(current\_pixel\_type)) + 4, or whatever; use the PF\_EffectWorld’s [rowbytes](#_bookmark232) instead. Never write outside the indicated region of a PF\_EffectWorld; this can corrupt cached image buffers that don't belong to you.

To test whether your effects are honoring the PF\_EffectWorld>[rowbytes](#_bookmark232), apply the

Grow Bounds effect *after* your effect. The output buffer will have larger rowbytes than the input (though it will still have the same logical size).

**BYTE ALIGNMENT**

The pixels in a [PF\_EffectWorld](#_bookmark226) are not guaranteed to be 16-byte-aligned. An effect may get a subregion of a larger PF\_EffectWorld. Users of Apple’s sample code for pixel

processing optimization, you have been warned.

**DEEPER COLOR**

Beyond 8-bit per channel color, After Effects supports 16 bit and 32-bit float per-channel color. Effects will never receive input and output worlds with differing bit depths, nor will they receive worlds with higher bit depth than they have claimed to be able to handle.

**ACCESSOR MACROS FOR OPAQUE (DATA TYPE) PIXELS**

Use the following macros to access the data within (opaque) PF\_PixelPtrs. It is,

emphatically, *not* safe to simply cast pointers of one type into another! To make it work at

all requires a cast, and there's nothing that prevents you from casting it incorrectly. We may change its implementation at a later date (at which time you’ll thank us for forcing this level of abstraction).

**TABLE 14: PF\_PIXELPTR ACCESSOR MACROS**

|  |  |
| --- | --- |
| **Macro** | **Purpose** |
| PF\_GET\_PIXEL\_DATA16 | Obtain a pointer to a 16-bpc pixel within the specified world. The returned pixel pointer will be NULL if the world is not 16-bpc. The second parameter is optional; if it is not NULL, the returned pixel will be an interpretation of the values in the passed-in pixel, as if it were in the specified PF\_EffectWorld.  PF\_GET\_PIXEL\_DATA16 (  PF\_EffectWorld \*wP, PF\_PixelPtr pP0,  PF\_Pixel16 \*\*outPP); |
| PF\_GET\_PIXEL\_DATA8 | Obtain a pointer to a 8-bpc pixel within the specified world. The returned pixel pointer will be NULL if the world is not 8- bpc. The second parameter is optional; if it is not NULL, the returned pixel will be an interpretation of the values in the passed-in pixel, as if it were in the specified PF\_EffectWorld.  PF\_GET\_PIXEL\_DATA8 (  PF\_EffectWorld \*wP, PF\_PixelPtr pP0,  PF\_Pixel8 \*\*outPP); |

[Think of](#_bookmark244) PF\_GET\_PIXEL\_DATA16 and [PF\_GET\_PIXEL\_DATA8](#_bookmark245) as safe (ahem) casting routines. The code required is actually very simple to get a PF\_Pixel16\* out of the PF\_EffectWorld output:

{

PF\_Pixel16 \*deep\_pixelP = NULL; PF\_Err err = PF\_Err\_NONE;

err = PF\_GET\_PIXEL\_DATA16(output, NULL, &deep\_pixelP);

}

This returns deep\_pixelP as NULL if the world does not have deep pixels. The second

parameter is not used very often and should be passed as NULL; pass a PF\_PixelPtr that is

*not* contained in a PF\_EffectWorld to coerce it to the depth of that PF\_EffectWorld).

### ERRORS

Always, always, *always* (always!) return a PF\_Err from main(). Plug-ins must pass all errors back to After Effects. It is vitally important that you pass any errors (returned to you by

callbacks and PICA suites) to After Effects, unless you’ve handled them. Be vigilant about returning the right error code, and disposing of any memory you’ve allocated. Really. We’re serious.

**TABLE 15: ERROR CODES**

|  |  |
| --- | --- |
| **Error** | **Meaning** |
| PF\_Err\_NONE | Success. |
| PF\_Err\_OUT\_OF\_MEMORY | Memory allocation failed. Note that RAM preview will cause this condition, so After Effects will be expecting to receive this error from your plug-in. |
| PF\_Err\_INTERNAL\_STRUCT\_DAMAGED | Problems using a data structure. |
| PF\_Err\_INVALID\_INDEX | Problems finding/using array member. |
| PF\_Err\_UNRECOGNIZED\_PARAM\_TYPE | Problem with parameter data. |
| PF\_Err\_INVALID\_CALLBACK | Problems accessing function through pointer. |
| PF\_Err\_BAD\_CALLBACK\_PARAM | Problems using a parameter passed to a callback. |
| PF\_Interrupt\_CANCEL | User cancelled rendering. |
| PF\_Err\_CANNOT\_PARSE\_KEYFRAME\_TEXT | Return this from PF\_Arbitrary\_SCAN\_FUNC when problems occur parsing the clipboard into keyframe data. |

**ERROR REPORTING POLICY**

After Effects has a consistent policy for error handling; follow it.

If you encounter an error in your plug-in’s code, report it to the user immediately, before returning from your plug-in to After Effects. After Effects considers errors from the

operating system, encountered during your plug-in’s execution, to be yours. If you get an error code back from one of our callback functions, pass it back to After Effects; we’ve

already reported it. Out-of-memory errors are never reported by After Effects. Error

reporting is always suppressed during RAM preview, and when After Effects is running in - noui mode.

To report an error from within a plug-in, set PF\_OutFlag\_DISPLAY\_ERROR\_MESSAGE, and describe the error in [PF\_OutData>return\_msg](#_bookmark140). Doing so will enter your error into the render log, and prevent system hangs in renders driven by a render engine or scripting.

**DIG IN!**

Now you have a basic understanding of effect plug-ins, and are ready to start experimenting with some real code. Go ahead and get started!

After getting the basics of your plug-in setup, you may have some questions about reuseable code, advanced functionality, and how to optimize your code to make it faster. To this end, After Effects exposes a tremendous amount of its internal functionality via function suites. By relying on After Effects code for utility functions, you should be able to get your image processing algorithms implemented quickly. This will discussed in the next chapter.

3 : EFFECT DETAILS

Now that we’ve covered the basics of effect plug-ins, we’ll cover some of the finer points to polish off your effect. Not every section will be relevant to every plug-in, so feel free to use the PDF document bookmarks to skip to the sections pertinent to your current project.

**FREE CODE == GOOD**

After Effects provides effect plug-ins with as much information and supporting code as

possible. Use our function suites and callbacks to obtain the value of parameters (including source footage) at different times. Use our memory allocation suite to avoid competing with the host for resources. Use our image processing suites to copy, fill, blend and convolve

images, and convert between color spaces. Obtain information about the masks applied to a layer. ANSI emulation and math utility suites are also provided, as well as information

about the application, user, serial number, and current drawing context.

Previous versions of After Effects have provided functions for many common tasks. As we moved to support deeper color, these were moved to function suites. Use the newer function suites whenever possible; things will just be better.

Using our function suites keeps your plug-in compact; you write and test less code. The

functions are tested, optimized, and used by our own plug-ins. The functions are distributed to multiple processors and take advantage of available hardware acceleration.

No, really, use the provided functions. Seriously.

### ACCESSING THE AFTER EFFECTS FUNCTION SUITES

If you are writing C++ code, accessing functions in our PICA function suites is a breeze, using the AEGP\_SuiteHandler, which automatically acquires the suite when needed, and disposes of it when done. Just instantiate the handler like so:

AEGP\_SuiteHandler suites(in\_data->pica\_basicP);

After that, you may make calls to any function in any suite, like so:

PF\_Handle infoH = suites.HandleSuite1()->host\_new\_handle(sizeof(MyStruct));

If you must use C code, then acquire and release the suites manually using the

PF\_Suite\_Helper utility files, as demonstrated in the Checkout sample project.

Behind the scenes, these both of these methods acquire PICA function suites using

AcquireSuite, a member function of the SPBasicSuite pointed to in [PF\_InData](#_bookmark111).

**SUITE VERSIONS**

WhizBangSuite1 may provide a Foobar() function which takes two arguments, and WhizBangSuite2>Foobar() may take three. Though each new version of a suite supercedes the old one, feel free to acquire multiple versions of the same suite; we never remove or alter previously shipped suites.

When unsure of the capabilities of the plug-in host (no third party host besides Premiere

supports PICA), attempt to acquire the latest version, and “fall back” to previous versions. If functionality you require isn’t available, warn the user, and return an error (or fall back on

other behavior when running in more “primitive” plug-in hosts). Note that support for these suites in other hosts of After Effects plug-ins is a maze of twisty caves and passages, all alike.

**THREADING**

Unless documented otherwise, assume that any function provided by our suites is not

thread-safe. For example, only your plug-in’s main thread should do anything that modifies the user interface.

## MEMORY ALLOCATION

Use After Effects for any memory allocations of significant size. For small allocations, you can use new and delete, but this is the exception, not the rule. In low-memory conditions

(such as during RAM preview), it’s very important that plug-ins deal gracefully with out-of- memory conditions, and not compete with After Effects for OS memory. By using our

memory allocation functions, After Effects can know when to free cached images, to avoid memory swapping. Failing to use our functions for sizable allocations can cause lock-ups, crashes, and tech support calls. Don’t do that.

If you’re wrapping existing C++ classes, create a base class that implements new and delete for that class and derive from it. To overload the STL, we don’t recommend you overload

global new and delete. Instead provide an allocator as part of the template definition.

Handles passed to you by After Effects are locked for you before you’re called, and unlocked once you return.

**TABLE 16: PF\_HANDLESUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| host\_new\_handle | Allocates a new handle. Replaces PF\_NEW\_HANDLE. PF\_Handle (\*host\_new\_handle)(  A\_HandleSize size); |
| host\_lock\_handle | Locks a handle. Replaces PF\_LOCK\_HANDLE. void\* (\*host\_lock\_handle)(  PF\_Handle pf\_handle); |
| host\_unlock\_handle | Unlocks a handle. Replaces PF\_UNLOCK\_HANDLE. void (\*host\_unlock\_handle)(  PF\_Handle pf\_handle); |
| host\_dispose\_handle | Frees a handle. Replaces PF\_DISPOSE\_HANDLE. void (\*host\_dispose\_handle)(  PF\_Handle pf\_handle); |
| host\_get\_handle\_size | Returns the size, in bytes, of the reallocatable block whose handle is passed in. Replaces PF\_GET\_HANDLE\_SIZE.  A\_HandleSize (\*host\_get\_handle\_size)( PF\_Handle pf\_handle); |
| host\_resize\_handle | Resizes a handle. Replaces PF\_RESIZE\_HANDLE. PF\_Err (\*host\_resize\_handle)(  A\_HandleSize new\_sizeL, PF\_Handle \*handlePH); |

**IMAGE BUFFER MANAGEMENT FUNCTIONS**

Use these functions to create and destroy [PF\_EffectWorld](#_bookmark227)s, and to find out their bit- depth.

**TABLE 17: PF\_WORLDSUITE2**

|  |  |
| --- | --- |
| **Function** | **Description** |
| PF\_NewWorld | Creates a new PF\_EffectWorld.  PF\_Err PF\_NewWorld(  PF\_ProgPtr effect\_ref,  A\_long widthL,  A\_long heightL,  PF\_Boolean clear\_pixB,  PF\_PixelFormat pixel\_format,  PF\_EffectWorld \*worldP); |
| PF\_DisposeWorld | Disposes of a PF\_EffectWorld.  PF\_Err PF\_DisposeWorld(  PF\_ProgPtr effect\_ref,  PF\_EffectWorld \*worldP); |
| PF\_GetPixelFormat | Get the pixel format for a given PF\_EffectWorld.  PF\_Err PF\_GetPixelFormat(  const PF\_EffectWorld \*worldP, PF\_PixelFormat \*pixel\_formatP);  pixel\_formatP can be: PF\_PixelFormat\_ARGB32 - standard 8-bit RGB PF\_PixelFormat\_ARGB64 - 16-bit RGB  PF\_PixelFormat\_ARGB128 - 32-bit floating point RGB |

### ITERATION SUITES

Effects often iterate over all pixels in an image, filtering each one. By taking advantage of

After Effects’ iteration suites, you make it possible for After Effects to sub-allocate your task to as many processors are present, taking advantage of hardware-specific acceleration. After Effects will also manage progress reporting and user cancellation automatically. Use these

suites! Make sure the pixel processing functions you pass to these iterator callbacks are re- entrant.

**TABLE 18: PF\_ITERATE8SUITE1, PF\_ITERATE16SUITE1, PF\_ITERATEFLOATSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| iterate | Iterates across pixels from a source image, alters them, and populates a destination image.  You may specify a rectangular region of pixels across which to iterate; if you don’t, After Effects will iterate over every overlapping pixel. You give a refcon, and the function is invoked with that refcon, plus the x and y coordinates of the current pixel, plus pointers to that pixel in the source and destination images. If you pass a NULL source, it will iterate over the dst. This function is quality independent.  Don’t depend upon the pixels being traversed in any particular order. The image may be subset to different CPUs, so consider all the parameters (except dst) to be read-only while After Effects is processing. This callback automatically includes progress and abort checking, so don’t do so in your pixel function.  iterate(  PF\_InData \*in\_data,  A\_long progress\_base,  A\_long progress\_final, PF\_EffectWorld \*src,  const PF\_Rect \*area,  void \*refcon,  PF\_Err (\*pix\_fn)(  void \*refcon, A\_long x, A\_long y, PF\_Pixel \*in, PF\_Pixel \*out),  PF\_EffectWorld \*dst); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| iterate\_origin | Lets you specify an offset from the input into the output. For example, if your output buffer is smaller than your input buffer, pass (in\_- data>output\_origin\_x, in\_data>output\_origin\_y) as the origin, and NULL for area, and this function will offset the src pixel pointer appropriately for your pixel function.  iterate\_origin(  PF\_InData \*in\_data,  A\_long progress\_base,  A\_long progress\_final, PF\_EffectWorld \*src,  const PF\_Rect \*area, const PF\_Point \*origin, void \*refcon,  PF\_Err (\*pix\_fn)(  void \*refcon, A\_long x, A\_long y, PF\_Pixel \*in, PF\_Pixel \*out),  PF\_EffectWorld \*dst); |
| iterate\_lut | PF\_Iterate8Suite only. Allows a Look-Up Table (LUT) to be passed for iteration; you can pass the same or different LUTs for each color channel. If no LUT is passed, an identity LUT is used.  iterate\_lut(  PF\_InData \*in\_data,  A\_long prog\_base,  A\_long prog\_final,  PF\_EffectWorld \*src,  const PF\_Rect \*area,  A\_u\_char \*a\_lut0,  A\_u\_char \*r\_lut0,  A\_u\_char \*g\_lut0,  A\_u\_char \*b\_lut0,  PF\_EffectWorld \*dst); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| iterate\_origin\_non\_ clip\_src | Allows for iteration across pixels outside the intersection of the source and destination layers. For these pixels, you will be passed a PF\_Pixel with values {0,0,0,0}.  iterate\_origin\_non\_clip\_src( PF\_InData \*in\_data,  A\_long progress\_base,  A\_long progress\_final, PF\_EffectWorld \*src,  const PF\_Rect \*area, const PF\_Point \*origin, void \*refcon,  PF\_Err (\*pix\_fn)(  void \*refcon, A\_long x, A\_long y, PF\_Pixel \*in, PF\_Pixel \*out),  PF\_EffectWorld \*dst); |
| iterate\_generic | PF\_Iterate8Suite only. If you want to do something once per available CPU, this is the function to use (pass PF\_Iterations\_ONCE\_PER\_PROCESSOR for  iterationsL). Only call abort and progress functions from thread index 0.  Note: You can iterate over more than pixels. Internally, we use it for row- based image processing, and for once-per-entity updates of complex sequence data.  iterate\_generic(  A\_long iterationsL,  void \*refconPV,  PF\_Err (\*fn\_func)(  void \*refconPV, A\_long thread\_idxL, A\_long i,  A\_long itrtL); |

### GRAPHICS UTILITY SUITES

After Effects exposes its internal transform and graphic utility routines through the following function suites.

These functions combine PF\_EffectWorlds in interesting ways. When you use these, you’re using the same code After Effects does internally.

**TABLE 19: PF\_WORLDTRANSFORMSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| composite\_rect | Composite a rectangle from one PF\_EffectWorld into another, using one of After Effects’ transfer modes.  PF\_Err composite\_rect ( PF\_ProgPtr effect\_ref,  PF\_Rect \*src\_rect,  A\_long src\_opacity, PF\_EffectWorld \*src\_world, A\_long dst\_x,  A\_long dst\_y,  PF\_Field field\_rdr,  PF\_XferMode xfer\_mode, PF\_EffectWorld \*dst);  field\_rdr can be upper, lower or both.  xfer\_mode is one of the following:  PF\_Xfer\_COPY PF\_Xfer\_BEHIND PF\_Xfer\_IN\_FRONT |
| blend | Blends two images, alpha-weighted. Does not deal with different-sized sources, though the destination may be either PF\_EffectWorld.  PF\_Err blend (  PF\_ProgPtr effect\_ref, const PF\_EffectWorld \*src1, const PF\_EffectWorld \*src2, PF\_Fixed ratio,  PF\_EffectWorld \*dst); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| convolve | Convolve an image with an arbitrary size kernel on each of the a, r, g, and b channels separately. You can specify a rectangle to convolve (for instance, the [extent\_hint](#_bookmark119)), or pass 0 to convolve the entire image. Do not use if the source *is* the destination. Describe the convolution using [*kernel flags*](#_bookmark269).  PF\_Err convolve( PF\_EffectWorld \*src,  const PF\_Rect \*area,  PF\_KernelFlags flags,  A\_long kernel\_size,  void \*a\_kernel,  void \*r\_kernel,  void \*g\_kernel,  void \*b\_kernel,  PF\_EffectWorld \*dst); |
| copy | Copies a region from one PF\_EffectWorld to another, preserving alpha (unlike the Mac OS CopyBits).  PF\_Err copy (  PF\_EffectWorld \*src,  PF\_EffectWorld \*dst,  PF\_Rect \*src\_r,  PF\_Rect \*dst\_r); |
| copy\_hq | A higher fidelity version of the above (using the same parameters). |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| transfer\_rect | Blends using a transfer mode, with an optional mask.  PF\_Err transfer\_rect (  PF\_ProgPtr effect\_ref,  PF\_Quality quality,  PF\_ModeFlags m\_flags,  PF\_Field field,  const PF\_Rect \*src\_rec, const PF\_EffectWorld \*src\_world, const PF\_CompositeMode \*comp\_mode, const PF\_MaskWorld \*mask\_world0, A\_long dest\_x,  A\_long dest\_y,  PF\_EffectWorld \*dst\_world); |
| transform\_world | Given a PF\_EffectWorld and a matrix (or array of matrices),  transforms and blends using an After Effects transfer mode, with an optional mask. The matrices pointer points to a matrix array used for motion-blur.  When is a transform not a transform? A Z-scale transform is not a  transform, unless the transformed layer is a parent of other layers that do not all lie in the z=0 plane.  PF\_Err transform\_world (  PF\_InData \*in\_data,  PF\_Quality quality,  PF\_ModeFlags m\_flags,  PF\_Field field,  const PF\_EffectWorld \*src\_world, const PF\_CompositeMode \*comp\_mode, const PF\_MaskWorld \*mask\_world0, const PF\_FloatMatrix \*matrices, A\_long num\_matrices,  Boolean src2dst\_matrix,  const PF\_Rect \*dest\_rect,  PF\_EffectWorld \*dst\_world); |

**KERNEL FLAGS**

Functions such as [convolve](#_bookmark267) or gaussian kernel work with kernels, or matrices of filter weight values. These matrices can be in any format. The kernel flags describe how the matrices

should be created and used. OR together any flags you need. The flags relevant to given

routines are documented along with the routine prototype.The first entry in the left column is always the default and has value 0.

**TABLE 20: KERNEL FLAGS**

|  |  |
| --- | --- |
| Kernel Flags | Indicates |
| PF\_KernelFlag\_2D PF\_KernelFlag\_1D | Specifies a one or two dimensional kernel. |
| PF\_KernelFlag\_UNNORMALIZED PF\_KernelFlag\_NORMALIZED | NORMALIZED equalizes the kernel;the volume under the kernel surface is the same as the volume under the covered area of pixels. |
| PF\_KernelFlag\_CLAMP PF\_KernelFlag\_NO\_CLAMP | CLAMP restricts values to the valid range for their data type. |
| PF\_KernelFlag\_USE\_LONG PF\_KernelFlag\_USE\_CHAR PF\_KernelFlag\_USE\_FIXED PF\_KernelFlag\_USE\_UNDEFINED | USE\_LONG defines the kernel as an array of longs valued from 0 to 255. USE\_CHAR defines the kernel as an array of unsigned chars from 0 to 255.  USE\_FIXED defines the kernel as an array of fixeds from 0 to 1. USE\_LONG is the only implemented flag. |
| PF\_KernelFlag\_HORIZONTAL PF\_KernelFlag\_VERTICAL | Specifies the direction of the convolution. |
| PF\_KernelFlag\_TRANSPARENT\_BORDERS PF\_KernelFlag\_REPLICATE\_BORDERS | Use REPLICATE\_BORDERS to replicate border pixels when sampling off the edge, use TRANSPARENT\_BORDERS to treat pixels off the edge as alpha zero (black).  REPLICATE\_BORDERS is not implemented and will be ignored. |
| PF\_KernelFlag\_STRAIGHT\_CONVOLVE PF\_KernelFlag\_ALPHA\_WEIGHT\_CONVOLVE | Use STRAIGHT\_CONVOLVE to indicate straight convolution, use ALPHA\_WEIGHT\_CONVOLVE to tell the convolution code to alpha-weight the contributions of pixels to the resulting convolved output.  ALPHA\_WEIGHT\_CONVOLVE is not implemented and will be ignored. |

**FILL ‘EM UP!**

The FillMatteSuite can be used to fill a PF\_EffectWorld, either with a specific color or premultiplied with an alpha value.

**TABLE 21: PF\_FILLMATTESUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| fill | Fills a rect with a color (or, if the color pointer is null, fills with black and alpha zero). If the rect is null, it fills the entire image.  PF\_Err fill (  PF\_ProgPtr effect\_ref, const PF\_Pixel \*color, const PF\_Rect \*dst\_rect, PF\_EffectWorld \*world); |
| fill16 | Same as fill, but takes a pointer to a PF\_Pixel16 color. |
| fill\_float | Takes a pointer to a PF\_PixelFloat color. |
| premultiply | Converts to (and from) r, g, and b color values pre-multiplied with black to represent the alpha channel. Quality independent. forward is used as a boolean; true means convert non-premultiplied to pre-multiplied, false mean un-pre-multiply.  PF\_Err premultiply (  A\_long forward, PF\_EffectWorld \*dst); |
| premultiply\_color | Converts to (and from) having r, g, and b color values premultiplied with any color to represent the alpha channel.  PF\_Err premultiply\_color ( PF\_ProgPtr effect\_ref, PF\_EffectWorld \*src, PF\_Pixel \*color,  A\_long forward, PF\_EffectWorld \*dst); |
| premultiply\_color16 | Same as above, but takes a pointer to a PF\_Pixel16 color. |
| premultiply\_color\_float | Takes a pointer to a PF\_PixelFloat color. |

**SAMPLING IMAGES**

Note: areas outside the bounds of the image being sampled are treated as zero alpha. For convenience, the functions from PF\_Sampling8Suite1, PF\_Sampling16Suite1, and

PF\_SamplingFloatSuite1 are all listed in this table.

**TABLE 22: PF\_SAMPLINGSUITE FUNCTIONS (MULTIPLE SUITES)**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| nn\_sample | Performs nearest neighbor sampling.  PF\_Err nn\_sample (  PF\_ProgPtr effect\_ref,  PF\_Fixed x,  PF\_Fixed y, const PF\_SampPB \*params, PF\_Pixel \*dst\_pixel ); |
| nn\_sample16 | Same as above, but takes a pointer to a PF\_Pixel16 dst\_pixel. |
| nn\_sample\_float | Takes a pointer to a PF\_PixelFloat dst\_pixel. |
| subpixel\_sample | Queries the appropriate alpha-weighted interpolation of colors at a non- integral point in a source image, in high quality. Nearest neighbor sampling is used in low quality. Because the sampling routine, if used, will typically be called many times, it is convenient to copy the function pointer out to the callbacks structure and into a register or onto the stack to speed up your inner loop. See the sample code for an example. NOTE: The sampling assumes that 0,0 is the center of the top left pixel.  PF\_Err subpixel\_sample ( PF\_ProgPtr effect\_ref,  PF\_Fixed x,  PF\_Fixed y, const PF\_SampPB \*params, PF\_Pixel \*dst\_pixel); |
| subpixel\_sample16 | Same as above, but takes a pointer to a PF\_Pixel16\* dst\_pixel. |
| subpixel\_sample\_float | Takes a pointer to a PF\_PixelFloat\* dst\_pixel. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| area\_sample | Use this to calculate the appropriate alpha weighted average of an axis- aligned non-integral rectangle of color in a source image, in high quality.  Nearest neighbor sampling is used in low quality. Because of overflow  issues, this can only average a maximum of a 256 x 256 pixel area (i.e. x and y radius < 128 pixels). NOTE: the sampling radius must be at least one in both x and y.  PF\_Err area\_sample (  PF\_ProgPtr effect\_ref,  PF\_Fixed x,  PF\_Fixed y, const PF\_SampPB \*params, PF\_Pixel \*dst\_pixel);  NOTE: Areas outside the boundaries of the layer are considered the same as zero alpha, for sampling purposes. |
| area\_sample16 | Same as above, but takes a PF\_Pixel16\* dst\_pixel. |

**TABLE 23: PF\_BATCHSAMPLINGSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| begin\_sampling | Your effect is going to perform some batch sampling; After Effects will perform setup tasks to optimize your sampling.  PF\_Err (\*begin\_sampling)( PF\_ProgPtr effect\_ref,  PF\_Quality qual,  PF\_ModeFlags mf,  PF\_SampPB \*params); |
| end\_sampling | Tells After Effects you’re done sampling.  PF\_Err (\*end\_sampling)( PF\_ProgPtr effect\_ref,  PF\_Quality qual,  PF\_ModeFlags mf,  PF\_SampPB \*params); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| get\_batch\_func | Obtains a pointer to After Effects’ batch sampling function (highly optimized).  PF\_Err (\*get\_batch\_func)( PF\_ProgPtr effect\_ref,  PF\_Quality quality,  PF\_ModeFlags mode\_flags, const PF\_SampPB \*params, PF\_BatchSampleFunc \*batch); |
| get\_batch\_func16 | Obtains a pointer to After Effects’ 16-bpc batch sampling function (also highly optimized).  PF\_Err (\*get\_batch\_func16)( PF\_ProgPtr effect\_ref,  PF\_Quality quality,  PF\_ModeFlags mode\_flags,  const PF\_SampPB \*params, PF\_BatchSample16Func \*batch); |

**DO THE MATH FOR ME**

Along with the variety of graphics utilities, we also provide a block of ANSI standard

routines so that plug-ins will not need to include other libraries to use standard functions. We give function pointers to a large number of math functions (trig functions, square root, logs, etc.).

Using our suite functions provides for some (application level) error handling, and prevents problems with including different versions of multiple “standard” libraries.

All functions return a double. All angles are expressed in radians, use PF\_RAD\_PER\_DEGREE

(a constant from AE\_EffectCB.h) to convert from degrees to radians if necessary.

**TABLE 24: PF\_ANSICALLBACKSSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| acos | Returns the arc cosine of x. Replaces PF\_ACOS. |
| asin | Returns the arc sine of x. Replaces PF\_ASIN. |
| atan | Returns the arc tangent of x. Replaces PF\_ATAN. |
| atan2 | Returns atan(y/x). Replaces PF\_ATAN2. |
| ceil | Returns the next integer above x. Replaces PF\_CEIL. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| cos | Returns the cosine of x. Replaces PF\_COS. |
| exp | Returns e to the power of x. Replaces PF\_EXP. |
| fabs | Returns the absolute value of x. Replaces PF\_FABS. |
| floor | Returns the closest integer below x. Replaces PF\_FLOOR. |
| fmod | Returns x modulus y. Replaces PF\_FMOD. |
| hypot | Returns the hypotenuse of x and y, which is sqrt(x\*x + y\*y). Replaces PF\_HYPOT. |
| log | Returns the natural log (ln) of x. Replaces PF\_LOG. |
| log10 | Returns the log (base 10) of x. Replaces PF\_LOG10. |
| pow | Returns x to the power of y. Replaces PF\_POW. |
| sin | Returns the sine of x. Replaces PF\_SIN. |
| sqrt | Returns the square root of x. Replaces PF\_SQRT. |
| tan | Returns the tangent of x. Replaces PF\_TAN. |
| *(while not strictly math functions, these emulate ANSI functionality)* | |
| sprintf | Emulates the C sprintf function. Replaces  PF\_SPRINTF. |
| strcpy | Emulates the C strcpy function. Replaces PF\_STRCPY. |

### INTERACTION CALLBACK FUNCTIONS

While the un-macro’d function pointers are provided in [PF\_InData](#_bookmark111), use the provided macros to access them. See how stringent we are about deprecating macro usage? Let’s let this be our little secret.

**TABLE 25: INTERACTION CALLBACKS**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_ADD\_PARAM | Enumerate your plug-in’s parameters to After Effects during  [*PF\_Cmd\_PARAM\_SETUP*](#_bookmark78), using multiple calls to this function.  Note: Failing to completely clear out a PF\_ParamDef prior to PF\_ADD\_PARAM() can cause many problems. Always use AEFX\_CLR\_STRUCT before adding parameters.  PF\_Err PF\_ADD\_PARAM (  PF\_InData \*in\_data, PF\_ParamIndex index, PF\_ParamDefPtr def);  We provide convenience macros for specific parameter types, in Utils/ Param\_Utils.h:  PF\_ADD\_COLOR, PF\_ADD\_ARBITRARY, PF\_ADD\_SLIDER, PF\_ADD\_FIXED, PF\_ADD\_FLOAT\_SLIDERX, PF\_ADD\_CHECKBOXX, PF\_ADD\_BUTTON, PF\_ADD\_ANGLE, PF\_ADD\_NULL, PF\_ADD\_LAYER, PF\_ADD\_255\_SLIDER, PF\_ADD\_PERCENT, PF\_ADD\_POINT, PF\_ADD\_POINT\_3D, PF\_ADD\_TOPICX, PF\_END\_TOPIC, PF\_ADD\_POPUPX,  PF\_ADD\_FLOAT\_SLIDERX\_DISABLED |
| PF\_ABORT | Returns non-zero if the user has cancelled; return that value to After Effects. Wrap your render routine in a “while abort has not been requested” while loop.  PF\_Err PF\_ABORT (PF\_InData \*in\_data); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_PROGRESS | Displays a progress bar during processing; current and total describe the percentage complete. Returns non-zero if you should suspend or abort your current processing; return that value to After Effects. Call once per scanline, unless your effect is very slow. If total is 0, PF\_ABORT is used instead (presenting the user with different choices).  PF\_Err PF\_PROGRESS (  PF\_InData \*in\_data, A\_long current,  A\_long total ); |
| PF\_CHECKOUT\_PARAM | Obtains parameter values, or the source video layer, at a specified time. After Effects makes caching decisions based on the checkout state of parameters.  Allocate a new [PF\_ParamDef](#_bookmark207) to hold the result; those passed to the plug- in are read-only. If you check out a layer parameter that’s set to <none>, the layer returned will be filled with zeros. Masks are not included with checked- out layers.  Do not check out layer parameters during UI event handling.  PF\_Err PF\_CHECKOUT\_PARAM (  PF\_InData \*in\_data, PF\_ParamIndex index, A\_long what\_time,  A\_long step,  A\_long time\_scale, PF\_ParamDef \*param);  If checking out the source layer, a deinterlaced frame will be returned. If you ask for the time that references the upper field, you will receive back the upper field with a filter used to generate the extra scanlines. For example, assuming line 0 and 2 are upper fields, and line 1 is a lower field, if you check out the upper fields, line 0 and 2 will be passed back directly from the source footage, and line 1 will be calculated by averaging lines 0 and 2. If you want to reassemble a full resolution source frame with both fields present, you can call PF\_CHECKOUT\_PARAM twice to get both fields, and reinterlace the footage.  What happens when checking out a layer at a time that is not frame-aligned? All items have essentially infinite time resolution, so when asking for a time at any value, AE renders the item at that time. For a composition, that involves interpolating all of the keyframes values to the subframe time. For footage, AE returns a full image that corresponds to the time asked, which is the nearest-to-left frame. If the user has frame-blending on that layer, an interpolated frame is generated. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_CHECKIN\_PARAM | Balance every PF\_CHECKOUT\_PARAM, with a PF\_CHECKIN\_PARAM. Not doing so causes dismal performance and leaks memory. Once checked in, the fields in the [PF\_ParamDef](#_bookmark207) will no longer be valid.  PF\_Err PF\_CHECKIN\_PARAM (  PF\_InData \*in\_data, PF\_ParamDef \*param ); |
| PF\_REGISTER\_UI | Register a custom user interface element. See [*Events*](#_bookmark425).  PF\_Err PF\_REGISTER\_UI (  PF\_InData \*in\_data, PF\_CustomUIInfo \*cust\_info ); |
| PF\_CHECKOUT\_LAYER\_AUDIO | Given an index, start\_time, duration, time\_scale, rate, bytes\_per\_sample, num\_channels, and fmt\_signed, After Effects will return a corresponding PF\_LayerAudio. After Effects will perform any necessary resampling.  PF\_Err PF\_CHECKOUT\_LAYER\_AUDIO (  PF\_InData \*in\_data, PF\_ParamIndex index, A\_long start\_time,  A\_long duration,  A\_u\_long time\_scale,  PF\_UFixed rate,  A\_long bytes\_per\_sample,  A\_long num\_channels,  A\_long fmt\_signed, PF\_LayerAudio \*audio); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_CHECKIN\_LAYER\_AUDIO | Balance all calls to PF\_CHECKOUT\_LAYER\_AUDIO, regardless of error conditions, with matching calls to PF\_CHECKIN\_LAYER\_AUDIO.  PF\_Err PF\_CHECKIN\_LAYER\_AUDIO (  PF\_InData \*in\_data, PF\_LayerAudio audio ); |
| PF\_GET\_AUDIO\_DATA | Returns information about the PF\_LayerAudio.  All the parameters after audio are optional; pass 0 for any value in which you aren’t interested. rate0 is unsigned, and fmt\_signed0 should be non-zero for signed, zero for unsigned. This callback is for visual effects that read audio information. To *alter* audio, write an audio filter.  PF\_Err PF\_GET\_AUDIO\_DATA (  PF\_InData \*in\_data, PF\_LayerAudio audio, PF\_SndSamplePtr \*data0, A\_long \*num\_samples0,  PF\_UFixed \*rate0,  A\_long \*bytes\_per\_sample0,  A\_long \*num\_channels0,  A\_long \*fmt\_signed0); |

**PARAMETER CHECKOUT VS. PARAM ZERO**

Effects are applied to an image in order from 0 to n within the Effect Control (and

Composition) panel. The output from effect[n-1] is the input ([param[0]](#_bookmark209)) of effect[n]. On the other hand, when a normal effect checks out a layer using [PF\_CHECKOUT\_PARAM](#_bookmark289), it

receives the raw (un-effected) source layer, regardless of its order. However, when a

[SmartFX](#_bookmark402) effect checks out its input parameter (params[0]), previous effects *are* applied.

**PARAMETER CHECKOUT BEHAVIOR**

Regardless of whether the layer in and out point have been trimmed, you will get valid

frames from the start of the source footage to the end, and then transparent before and after that.

Layer params with a lower frame rate than the composition in which they’re checked out are only refreshed as often as necessitated by the lower frame rate. A 10fps layer checked out in a 30fps composition will only need to be refreshed every third frame. if your effect wants to change it’s output every frame despite the static input layer, you’d need to set [PF\_Outflag\_NON\_PARAM\_VARY](#_bookmark147).

When an effect checks out a continuously-rasterized Adobe Illustrator layer, After Effects renders the Illustrator layer with geometrics applied, in a composition-sized buffer.

**PARAMETER CHECKOUT AND RE-ENTRANCY**

Plug-ins that check out layers at different times can generate re-entrant behavior. Consider an instance where the Checkout sample plug-in is applied to a layer in composition B, and B is pre-composed into composition A where Checkout is applied to it as well. When

composition A is rendered, Checkout[A] will be sent *PF\_Cmd\_RENDER*, during which it checks out a layer (composition B) from a time other than the current time. In order to

provide that checked-out layer, After Effects sends *PF\_Cmd\_RENDER* to Checkout[B]. Presto, recursion!

If you’re going to check out parameters, your effects must handle re-entrant render requests appropriately. Don’t use globals, or read or write static variables...but you weren’t going to anyway, right?

**PROGRESS DURING ITERATION**

After Effects strives to be as responsive as possible to user interaction, even while rendering. Do the same through appropriate use of PF\_ITERATE(). For example, perhaps you’re using a PF\_ITERATE’d function three times during your response to [*PF\_Cmd\_RENDER*](#_bookmark89). In this case, you’d start off with:

lines\_per\_iterateL = in\_data>extent\_hint.top - in\_data>extent\_hint.bottom;

total\_linesL = 3 \* lines\_per\_iterateL; lines\_so\_farL = 0;

After each iteration, you’d add the already-completed lines to the current position.

suites.iterate8suite()>iterate( lines\_so\_farL,

total\_linesL, input\_worldP,

&output>extent\_hint, refcon, WhizBangPreProcessFun, output\_worldP);

lines\_so\_farL += lines\_per\_iterateL; ERR(PF\_PROGRESS(lines\_so\_farL, total\_linesL));

suites.iterate8suite()>iterate( lines\_so\_farL,

total\_linesL, input\_worldP,

&output>extent\_hint, refcon, WhizBangRenderFunc, output\_worldP);

lines\_so\_far += lines\_per\_iterateL; ERR(PF\_PROGRESS(lines\_so\_farL, total\_linesL)); suites.iterate8suite()>iterate( lines\_so\_farL,

total\_linesL,

input\_worldP,

&output>extent\_hint, refcon, WhizBangPostProcessFunc, output\_worldP);

ERR(PF\_PROGRESS(lines\_so\_farL, total\_linesL));

## PIXEL ASPECT RATIO

Effects must respond correctly to footage with non-square pixels, and non-uniform

downsampling factors. Even different layer parameters can have different pixel aspect ratios! Doing so isn’t difficult once you understand the concepts involved.

Simple effects needn’t do any work to match up [point parameters](#_bookmark196) to the actual pixels in the output. Point parameters are given to the effect scaled for downsample factor and pixel

aspect ratio; they are in the coordinate system of the input buffer. This provides an implicit “pixel coordinate system.” This coordinate system is handy and easy to understand. But

effects that use absolute pixel measurements or geometry must take a deeper look at the relationship between the input buffer and the final rendered image.

**DON’T ASSUME PIXELS ARE SQUARE, OR 1-TO-1**

First, it is not necessarily a square coordinate system, due to both pixel aspect ratio and non- uniform downsample factor. The final rendered image can be stretched or squashed

horizontally, relative to the pixels your effect processes. Circles will appear as ellipses,

squares as rectangles. The distance between two points varies based on their angle in this coordinate system; anything rotated in this system is skewed, in the final output.

Second, even if it *is* a square coordinate system, it's not necessarily the same size as the final output. This means that any slider which defines a size in pixels will be a problem when the image is rendered downsampled; the width of anti-aliasing filters changes based on

downsample factor.

Sometimes these issues aren't a problem. Any effect that colors pixels based solely on a linear function of the x and y coordinates need not bother with pixel aspect ratio and downsample factor at all. Staying in the input coordinate space is an option, though you must account for pixel aspect ratio and downsample factor elsewhere.

Suppose you're writing a particle system effect that sprays textured sprites from a source

position defined by an effect control point. Using pixel coordinates to represent the particle positions seems fine (as long as the particles don't have to rotate around a point), but when you go to actually *render* the particle textures, you'll have to scale them by pixel aspect ratio and downsample factor.

If an effect already has coordinate transformation machinery in its pipeline, there's an alternative that's often simpler. Many algorithms require some sort of coordinate

transformation; using matrices to set up a transformation, for example. But there are other easily adaptable algorithms, for example a texture generation effect that computes the value of each pixel based solely on its position. In this case, the code must take the raw pixel

position and account for pixel aspect ratio and downsample factor.

**SUGGESTED APPROACH**

The simplest way to get all of this right is to work entirely in full resolution square coordinates, then scale by downsample factor and pixel aspect ratio as a final output

transformation. Since point parameters are always reported in input buffer coordinates,

convert them to full-resolution square coordinates before use. With this approach you don't need to worry about sliders which define a size in pixels; just interpret them as defining size in full-resolution vertical pixels.

1. When getting your point parameters, go immediately to floating point and a full resolution square pixel system, like this:

x \*= in\_data>pixel\_aspect\_ratio.num / (float)in\_data>pixel\_aspect\_ratio.den;

1. \*= in\_data>downsample\_x.den / (float)in\_data>downsample\_x.num;
2. \*= in\_data>downsample\_y.den / (float)in\_data>downsample\_y.num;
3. Perform all setup (define transformation matrices, generate coordinates for later scan

conversion, compute values based on the distance between points, rotating things, et cetera) in this coordinate space. Note that you're not actually dealing with pixels in this stage; you're just manipulating coordinates or coordinate transformations.

1. To go back to a coordinate system that corresponds directly to the pixels of the output buffer, undo the transformations from step one. Do this as late as possible, so as little code

as possible needs to deal with this non-square space. If you're using matrices, this would be a final output transformation. For an effect which renders something based on the coordinate of each pixel, iterate over the output pixels and convert pixel coordinates to square

coordinates before doing any processing for that pixel.

This may seem like extra work, but most reasonably complex effects like this have a

coordinate transformation step anyway; and if they don't, they still need one to handle pixel aspect ratio and downsample factor correctly.

**APPLYING USER INPUT IN PIXELS**

After Effects does all of its stretching horizontally so as to not to introduce unnecessary field interpolations; when pixels are used as a unit, we think of them as vertical pixels.

**TEST TEST TEST!**

Test at 1/2, 1/4, and custom resolutions and compare the output. Use an anamorphic (2:1) pixel aspect ratio composition to track down bugs in pixel aspect ratio handling (it really

makes them obvious), and be sure to test with different horizontal and vertical downsample factors.

Some developers have reported problems with the downsample factors provided by some “After Effects compatible” plug-in hosts being zero. Check for zero before dividing.

**PARAMETERS AND FLOATING POINT VALUES**

We have something to admit to you; for years, even though we’ve given you 8 bit color

values, we’ve internally used floating point representations behind your back. That’s right, even with over-bright colors, we’d only ever tell you ‘255, 255, 255’. Yeah, right. Well, we

can’t live the lie any longer! Given a color parameter (passed to you by After Effects in your

effect’s parameter array), this function returns a floating point representation, including any high dynamic range component.

**TABLE 26: PF\_COLORPARAMSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetFloatingPoint ColorFromColorDef | PF\_Err PF\_GetFloatingPointColorFromColorDef( PF\_ProgPtr effect\_ref,  const PF\_ParamDef \*color\_defP, PF\_PixelFloat \*fp\_colorP); |

We also provide a way to get floating point values for point parameters.

**TABLE 27: PF\_POINTPARAMSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetFloatingPoint ValueFromPointDef | PF\_Err PF\_GetFloatingPointValueFromPointDef( PF\_ProgPtr effect\_ref,  const PF\_ParamDef \*point\_defP, A\_FloatPoint \*fp\_pointP); |

New in CS6.0.2, we now provide a way to get floating point values for angle parameters.

**TABLE 28: PF\_ANGLEPARAMSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetFloatingPoint ValueFromAngleDef | PF\_Err PF\_GetFloatingPointValueFromAngleDef( PF\_ProgPtr effect\_ref,  const PF\_ParamDef \*angle\_defP, A\_FloatLong \*fp\_valueP); |

### PARAMETER SUPERVISION

Supervision means dynamically changing the values of some parameters based on the values of others. To supervise a parameter, set [PF\_ParamFlag\_SUPERVISE](#_bookmark219) before adding it

during *PF\_Cmd\_PARAMS\_SETUP*. Whenever it is changed, you will receive

[*PF\_Cmd\_USER\_CHANGED\_PARAM*](#_bookmark102). The index (into the plug-in’s parameter array) of the changed parameter is sent in the PF\_UserChangedParamExtra (extra) param. During *PF\_Cmd\_USER\_CHANGED\_PARAM*, you may change the values *and* appearance of any of your parameters.

**UPDATING PARAMETER UI**

If you set PF\_ParamFlag\_SUPERVISE on any parameter, After Effects will send you *PF\_Cmd\_UPDATE\_PARAMS\_UI*, just as if you had set PF\_OutFlag\_SEND\_UPDATE\_PARAMS\_UI.

During *PF\_Cmd\_UPDATE\_PARAMS\_UI*, you may only change the appearance and enable state of parameters. Use [PF\_UpdateParamUI](#_bookmark321)() from [PF\_ParamUtilsSuite](#_bookmark319) to update

the UI, passing it a *copy* of the parameter you wish to modify. Do *not* attempt to modify the original. It is not necessary to set PF\_OutFlag\_REFRESH\_UI; PF\_UpdateParamUI() handles that for you. Note also that this is the only way to update the UI of PF\_PUI\_STD\_CONTROL\_ONLY parameters.

**UPDATING PARAMETER VALUES**

A parameter’s value (not just UI) can be modified during

[*PF\_Cmd\_USER\_CHANGED\_PARAM*](#_bookmark102) and during [*PF\_Cmd\_EVENT*](#_bookmark100)

(*PF\_Event\_DO\_CLICK*, *PF\_Event\_DRAG*, & *PF\_Event\_KEYDOWN*). After Effects will not honor changes made at other times.

When changing parameter *values* (and not just the UI), modify the original parameter, and set PF\_Paramdef.uu.change\_flags to PF\_ChangeFlag\_CHANGED\_VALUE. This change will be also update the UI, and will be undoable by the user. Note that PF\_ChangeFlag\_CHANGED\_VALUE isn’t supported for layer parameters.

This suite is provided to give effect plug-ins some access to their parameter streams, without requiring AEGP suite usage. At least some of these functions are provided by several third- party hosts. These functions are especially handy for effects with supervised parameters.

**TABLE 29: PF\_PARAMUTILSSUITE3**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_UpdateParamUI | PF\_UpdateParamUI(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index, const PF\_ParamDef \*defP);  Force After Effects to refresh the parameter’s UI, in the effect controls palette.  Starting in CC 2014, After Effects will now honor a change to a custom UI height. Simply change the ui\_height of your custom UI  PF\_ParamDef and then call PF\_UpdateParamUI. The effect's custom UI height will be updated in the Effect Control Window.  Starting in CS6, when a plug-in disables a parameter, we now save that state in the UI flags so that the plug-in can check that flag in the future to see if it is disabled.  NOTE: Never pass param[0] to this function. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetCurrentState | PF\_GetCurrentState(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index,  const A\_Time \*startPT0,  const A\_Time \*durationPT0,  PF\_State \*stateP);  This API, combined with PF\_AreStatesIdentical below, lets you determine if a set of inputs (either layers, other properties, or both) are different between when you first called PF\_GetCurrentState and a current call, so it can be used for caching. You can specify a range of time to consider or all of time.  Updated in CS6 to add param\_index, startPT0, and durationPT0. Pre-defined constants for param\_index are as follows:  PF\_ParamIndex\_CHECK\_ALL - check every parameter, including every layer referred to by a layer parameter.  PF\_ParamIndex\_CHECK\_ALL\_EXCEPT\_LAYER\_PARAMS -  omit all layers. Pass a specific layer parameter index to include that as the only layer parameter tested.  PF\_ParamIndex\_CHECK\_ALL\_HONOR\_EXCLUDE - Similar to CHECK\_ALL, but honor PF\_ParamFlag\_EXCLUDE\_FROM\_HAVE\_INPUTS\_CHANGED.  Passing in NULL for both start and duration indicates all time. For effects that do simulation across time and therefore set PF\_OutFlag2\_AUTOMATIC\_WIDE\_TIME\_INPUT, when you ask about a time range, it will be expanded to include any times needed to produce that range.  Populates a PF\_State, an opaque data type used as a receipt for the current state of the effect’s parameters (the PF\_State is used in our internal frame caching database). |
| PF\_AreStatesIdentical | PF\_AreStatesIdentical(  PF\_ProgPtr effect\_ref, const PF\_State \*state1P, const PF\_State \*state2P, A\_Boolean \*samePB);  New in CS6. Compare two different states, retrieved using  PF\_GetCurrentState, above. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_HasParamChanged | No longer supported in PFParamUtilsSuite3.  PF\_HasParamChanged(  PF\_ProgPtr effect\_ref,  const PF\_State \*stateP,  PF\_ParamIndex param\_index,  PF\_Boolean \*changedPB);  Given a PF\_State, passes back true if any of the tested parameters differ from the saved state. Contrary to the name, the call does not  provide a way to test a single parameter. At a minimum, all non-layer parameters will be tested. For finer granularity to test a specific set of parameters, use PF\_HaveInputsChangedOverTimeSpan  below instead.  Pre-defined constants for param\_index are as follows:  PF\_ParamIndex\_CHECK\_ALL - check every parameter, including every layer referred to by a layer parameter.  PF\_ParamIndex\_CHECK\_ALL\_EXCEPT\_LAYER\_PARAMS -  omit all layers. Pass a specific layer parameter index to include that as the only layer parameter tested. |
| PF\_HaveInputsChanged OverTimeSpan | No longer supported in PFParamUtilsSuite3. Use PF\_AreStatesIdentical() instead. |
| PF\_IsIdenticalCheckout | PF\_IsIdenticalCheckout(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index,  A\_long what\_time1,  A\_long time\_step1,  A\_u\_long time\_scale1,  A\_long what\_time2,  A\_long time\_step2,  A\_u\_long time\_scale2,  PF\_Boolean \*identicalPB);  Returns TRUE if a parameter’s value is the same at the two passed times. Note: the times need not be contiguous; there could be  different intervening values. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_FindKeyframeTime | PF\_FindKeyframeTime(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index,  A\_long what\_time,  A\_u\_long time\_scale,  PF\_TimeDir time\_dir,  PF\_Boolean \*foundPB,  PF\_KeyIndex \*key\_indexP0,  A\_long \*key\_timeP0,  A\_u\_long \*key\_timescaleP0);  Searches (in the specified direction) for the next keyframe in the parameter’s stream. The last three parameters are optional. |
| PF\_GetKeyframeCount | PF\_GetKeyframeCount(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index,  PF\_KeyIndex \*key\_countP);  Returns the number of keyframes in the parameter’s stream. |
| PF\_CheckoutKeyframe | PF\_CheckoutKeyframe(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index,  PF\_KeyIndex key\_index,  A\_long \*key\_timeP0,  A\_u\_long \*key\_timescaleP0,  PF\_ParamDef \*paramP0);  Checks a keyframe for the specified parameter out of our keyframe database. param\_index is zero-based. You can request time, timescale, or neither; useful if you’re performing your own motion blur. |
| PF\_CheckinKeyframe | PF\_CheckinKeyframe(  PF\_ProgPtr effect\_ref,  PF\_ParamDef \*paramP);  All calls to PF\_CheckoutKeyframe must be balanced with this check-in, or pain will ensue. |
| PF\_KeyIndexToTime | PF\_KeyIndexToTime(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index,  PF\_KeyIndex key\_indexP,  A\_long \*key\_timeP,  A\_u\_long \*key\_timescaleP);  Returns the time (and timescale) of the specified keyframe. |

### GLOBAL, SEQUENCE, AND FRAME DATA

After Effects allows plug-ins to store data at three scopes; global, sequence, and frame.

Consider carefully where you store information; choosing poorly can impact performance, or make your plug-in confusing to the user.

Use global data for information common to all instances of the effect: static variables and data, bitmaps, pointers to other DLLs or external applications.

Store anything specific to this instance of your plug-in (UI settings, text strings, and any custom data not stored in parameters) in sequence data, Use After Effects’ memory

allocation functions.

Frame data is used for information specific to rendering a given frame. This has fallen into disuse, as most machines are capable of loading an entire frame into memory at a time. Of course, your IMAX-generating users will still appreciate any optimizations you can make.

**PERSISTENCE**

After Effects saves sequence data in the project file, but not global or frame data. Pointers within sequence data which point to external data are, in all likelihood, invalid upon re-

opening the project, and must be re-connected. We call this process “flattening” and “un- flattening” the sequence data.

**VALIDATING SEQUENCE DATA**

Careful sequence data validation is important for effects that do simulation across time, where frame N is dependent on frame N-1, and you use a cache of calculated data in your

sequence data. If a parameter is changed, certain calculated data may no longer be valid, but it would also be wasteful to blindly recalculate everything after every change.

When asked to render frame N, assuming you have your cached data calculated up to frame N-1, call [PF\_GetCurrentState()](#_bookmark323) / [PF\_AreStatesIdentical()](#_bookmark325) to see if the cache of calculated data is still valid given the current parameter settings. The state of all parameters (except those with [PF\_ParamFlag\_EXCLUDE\_FROM\_HAVE\_INPUTS\_CHANGED](#_bookmark223) set),

including layer parameters (including [param[0]](#_bookmark209)) are checked over the passed time span. This is done efficiently, as the change tracking is done with timestamps.

If the inputs have not changed, you can safely use your cache, AND the internal caching system will assume that you have a temporal dependency on the passed range. So if

something changes upstream, the host’s caches will be properly invalidated automatically.

To test that it is working, apply your effect with one parameter keyframed on every frame. RAM Preview to fill the cache, then change one of the keyframes. The related frame and all dependent frames (e.g. later frames, in the case of a simulation) should lose their cache

marks and require re-rendering. Similarly, upstream changes to sources of layer parameters should cause time-selective invalidation of the cache.

**FLATTENED AND UNFLATTENED SEQUENCE DATA**

If your sequence data references external memory (in pointers or handles), you must flatten and unflatten your data for disk-safe storage. This is analogous to creating your own

miniature file format.

Upon receiving [*PF\_Cmd\_SEQUENCE\_FLATTEN*](#_bookmark84)*,* put data referenced by pointers into one contiguous block from which you can later recover the old structure. If your sequence data

contains a pointer to a long, allocate 4 bytes in which to store the flattened data. You must handle platform-specific byte ordering.

Remember, your users (the ones who bought two copies of your plug-in, anyway) may want the same project to work on Mac OS and Windows. After Effects sends

[*PF\_Cmd\_SEQUENCE\_RESETUP*](#_bookmark82) when the data is reloaded, for either flat or unflat data. Use a flag at a common offset within both structures to indicate the data’s state.

typedef struct {

A\_char\* messageZ;

PF\_FpLong big\_numF;

void\* temp\_storage;

} non\_flat\_data;

typedef struct {

char message[256];

PF\_FpLong big\_numF; A\_Boolean big\_endianB;

} flat\_data;

**RESIZING SEQUENCE DATA**

During [*PF\_Cmd\_SEQUENCE\_SETUP*](#_bookmark80), allocate a handle for data specific to this instance of your effect. you may modify the contents, but not the size, of the sequence data during any

selector. You may resize the sequence data handle only during the following selectors:

*PF\_Cmd\_AUDIO\_SETUP*

*PF\_Cmd\_AUDIO\_SETDOWN PF\_Cmd\_FRAME\_SETUP*

*PF\_Cmd\_FRAME\_SETDOWN PF\_Cmd\_AUDIO\_RENDER*

*PF\_Cmd\_RENDER*

*PF\_Cmd\_SEQUENCE\_SETUP (duh) PF\_Cmd\_SEQUENCE\_SETDOWN PF\_Cmd\_SEQUENCE\_FLATTEN*

*PF\_Cmd\_SEQUENCE\_RESETUP*

*PF\_Cmd\_DO\_DIALOG*

### ARBITRARY DATA PARAMETERS

Some values are not adequately represented by After Effects existing parameter types. You can create and register any data for interpolation by After Effects, by creating parameters of arbitrary data type, or “arb data”. You can rely on our interpolation engine and parameter management, without having to force your data into a pre-defined parameter type.

We’ve created a new messaging structure for custom data types, which are easily

conceptualized as member (and friend) functions of a C++ class. You must respond to all selectors detailed here if you use arb data.

These functions deal with custom data structure management. Your arb data will be

unloaded and reloaded at the user’s whim; provide disk-safe flatten and unflatten functions.

**TABLE 30: ARBITRARY DATA SELECTORS**

|  |  |
| --- | --- |
| **Selector** | **Response** |
| *PF\_Arbitrary\_NEW\_FUNC* | Allocate, populate, and return a handle to a new instance of your arb data. |
| *PF\_Arbitrary\_DISPOSE\_FUNC* | Free and destroy an instance of your arbitrary data type. |
| *PF\_Arbitrary\_COPY\_FUNC* | Make a copy of an existing instance. You will be passed two handles, but only the source handle contains a valid instance.  You must create a new instance, copy the values from the  source, and put it in the destination handle. If you are passed a NULL handle, create a default instance of your arb data. |
| *PF\_Arbitrary\_FLAT\_SIZE\_FUNC* | You’ll be passed a handle to an instance of your data type, and a variable in which you return the size of a flattened version of that instance. |
| *PF\_Arbitrary\_FLATTEN\_FUNC* | Flatten the instance you’re passed, and place it in the supplied buffer. The buffer will be the size you reported in response to PF\_Arbitrary\_FLAT\_SIZE\_FUNC. |
| *PF\_Arbitrary\_UNFLATTEN\_FUNC* | Unpack the buffer into an instance of your arbitrary data type, and put in the handle which you’ve been passed. |

**TABLE 30: ARBITRARY DATA SELECTORS**

|  |  |
| --- | --- |
| **Selector** | **Response** |
| *PF\_Arbitrary\_INTERP\_FUNC* | Your interpolation function is passed three handles to  instances of your arbitrary data type; one containing initial values (0), one final values (1), and a third to hold your  interpolated data (somewhere between 0 and 1). You are also passed a float indicating where, between 0 and 1, your  interpreted value should be.  Allocate an instance and fill it with interpolated data. Then put the interpolated instance into the handle you’ve been  passed. The velocity curves have already been accounted for when the normalized time value was calculated.  NOTE: Never check out parameters if the  [in\_data>effect\_ref](#_bookmark113) is NULL. |
| *PF\_Arbitrary\_COMPARE\_FUNC* | You are passed two instances of your arbitrary data, and a  pointer to a comparison result. Populate the result with one of the values for PF\_ArbCompareResult (see AE\_Effect.h) to indicate whether the first was equal to, less than, more than, or simply not equal to the second. |
| *PF\_Arbitrary\_PRINT\_SIZE\_FUNC* | Indicate the buffer size you require for printing your  parameter’s current values by setting print\_sizePLu (member of print\_size\_func\_params, part of the PF\_ArbParamsExtra structure). |
| *PF\_Arbitrary\_PRINT\_FUNC* | Format your arbitrary data for text-based export, and copy the result to the buffer. This can be as elaborate as you would like.  Your plug-in should emulate the cut-and-paste behavior for pasting text representations of parameter settings (into a  Microsoft Excel spreadsheet, for example) displayed by the plug-ins shipped with After Effects. You have a great deal of flexibility in how you format your output. |
| *PF\_Arbitrary\_SCAN\_FUNC* | Given a buffer of text data (often from the system clipboard), parse it into your arbitrary data format. |

**IMPLEMENTING ARBITRARY DATA**

In addition to the normal command and event selector, arb data requires another set of host interaction. This is transparent for other parameter types, as After Effects manages their

representing data. Writing an arb data plug-in will give you insight into the vast amount of parameter management After Effects performs, and the sequence in which those managing

actions occur. It may even cause you to rethink your implementation, and use the parameter types After Effects manages *for* you.

Instantiate your arb data (using After Effects’ memory allocation functions, of course) and point ParamDef.u.arb\_d.dephault at it. Populate it with appropriate default values. No value variable is required to set up the parameter; zero it out for safety’s sake.

In your plug-in’s entry function, include a case for handling

[*PF\_Cmd\_ARBITRARY\_CALLBACK*](#_bookmark105). Invoke a secondary event handler, HandleArbitrary. It receives a PF\_ArbParamsExtra in extra, which in turn contains a PF\_FunctionSelector identifying the command sent.

Perhaps After Effects has sent PF\_Cmd\_ARBITRARY\_CALLBACK and the PF\_FunctionSelector is [PF\_Arbitrary\_COPY\_FUNC](#_bookmark337). Pointers to a source and destination Arb are provided in PF\_ArbParamsExtra.copy\_func\_params. Allocate a new Arb, and point dest\_arbPH at it. If src\_arbH is NULL, create a default Arb for dest\_arbPH.

The user may select the arb’s keyframe data in the Timeline panel, copy it, then switch to another application. You will be sent a PF\_Arbitrary\_PRINT\_SIZE\_FUNC; set the size of your output buffer by setting print\_sizePLu in the PF\_ArbParamsExtra. You’ll then receive PF\_Arbitrary\_PRINT\_FUNC; populate the print\_bufferPC output buffer with a textual

representation of the Arb(s) in question.

Users may paste keyframe data into your Arb’s timeline. You will receive PF\_Arbitrary\_SCAN\_FUNC. Create an Arb based on the contents of the character buffer handed to you (its size is indicated in print\_sizeLu).

**ARBITRARY DATA? RE-ENTRANCY.**

Your plug-in code *must* be recursively re-entrant to support custom data types, since it could be called by After Effects for numerous reasons. Your plug-in could check out a layer that, in turn, depends on another instance of your effect. Your plug-in’s arbitrary data handling code will be triggered by your attempt to check out a (seemingly) unrelated layer. Watch out for

calls to C run-time libraries that rely on static values accessed through global variables. If you’re not prepared for this eventuality, you’ll hang After Effects, and users will curse and punch their monitors.

**WHEN NOT TO ACCESS ARBITRARY PARAMETERS**

If in\_data>effect\_ref is NULL, do not check out arbitrary parameters.

**CHANGES DURING DIALOGS**

After Effects ignores any changes made to arbitrary data parameters during

*PF\_Cmd\_DO\_DIALOG*. This is by design; changes made during the display of the options

dialog affect the entire effect stream, not just the arbitrary parameter at a given time. If you must alter your arb’s behavior based on these changes, save that information in sequence

data and apply it later, often during *PF\_Cmd\_USER\_CHANGED\_PARAM*.

### USEFUL UTILITY FUNCTIONS

**PF\_EFFECTUISUITE**

Although not strictly concerned with parameters, this suite can change the name of the options button.

**TABLE 31: PF\_EFFECTUISUITE**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_SetOptionsButtonName | Changes the text on the options button in the effect controls palette. NOTE: This must be called during [*PF\_Cmd\_PARAM\_SETUP*](#_bookmark78).  PF\_SetOptionsButtonName( PF\_ProgPtr effect\_ref,  const A\_char \*nameZ);  nameZ may be up to A\_char[31] in length. |

**PF\_APPSUITE**

Roughly 437 years ago, when we released After Effects 5.0, we published some useful utility callbacks in PF\_AppSuite. They’re as useful today as they were then. After Effects has user- controllable UI brightness. In addition to the [PF\_EffectCustomUIOverlayThemeSuite](#_bookmark496) for custom UI in effects, use these calls to integrate seamlessly into the After Effects UI.

What better way to shame someone into purchasing a copy of your plug-in than by putting

their personal information into a watermark, eh? Or set the cursor to add mask vertices, just to confuse people? Heh heh heh. But that would be wrong.

**TABLE 32: PF\_APPSUITE5**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_AppGetBgColor | Retrieves the current background color.  PF\_AppGetBgColor(  PF\_App\_Color \*bg\_colorP); |
| PF\_AppGetColor | Retrieves the color for the specified UI element. See  AE\_EffectSuites.h for a complete enumeration of available PF\_App\_Color values; basically any color in After Effects’ UI can be retrieved.  CC adds several new PF\_App\_ColorType enum values for new  elements that can be queried. Note that in CS6, the color definitions are off from FILL\_LIGHT downward. Use following psuedocode for CS6 only:  GetColor(enum e)  {  if host\_is\_CS6 and e >= FILL\_LIGHT e += 3  call real GetColor  }  PF\_AppGetColor(  PF\_App\_ColorType color\_type, PF\_App\_Color \*app\_colorP); |
| PF\_AppGetLanguage | New in CC. Retrieves the active displayed language of AE UI so plug- in can match. Here are the possible language codes as of CC:  Chinese - zh\_CN English - en\_US French - fr\_FR  German - de\_DE Italian - it\_IT Japanese - ja\_JP Korean - ko\_KR Spanish - es\_ES  PF\_AppGetLanguage(  A\_char \*lang\_tagZ); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetPersonalInfo | Retrieves the user’s registration information.  PF\_GetPersonalInfo( PF\_AppPersonalTextInfo \*ptiP);  typedef struct PF\_AppPersonalTextInfo { A\_char name[PF\_APP\_MAX\_PERS\_LEN + 1]; A\_char org[PF\_APP\_MAX\_PERS\_LEN + 1]; A\_char serial\_str[PF\_APP\_MAX\_PERS\_LEN+1];  } PF\_AppPersonalTextInfo; |
| PF\_GetFontStyleSheet | Retrieves font style sheet information for the fonts used in After  Effects’ UI. Trivia: The fonts used in After Effects’ UI are Tahoma on Windows and Lucida Grande on Mac OS X.  PF\_GetFontStyleSheet( PF\_FontStyleSheet sheet, PF\_FontName \*font\_nameP0,  A\_short \*font\_numPS0,  A\_short \*sizePS0,  A\_short \*stylePS0); |
| PF\_SetCursor | Sets the cursor to any of After Effects’ cursors. See AE\_EffectUI.h for a complete enumeration. Set to PF\_Cursor\_NONE to allow After  Effects to set the cursor. Set to PF\_Cursor\_CUSTOM if you’ve used OS-specific calls to change the cursor (After Effects will honor your  changes).  PF\_SetCursor(  PF\_CursorType cursor); |
| PF\_IsRenderEngine | Returns TRUE if After Effects is running in watched folder mode, or is a render engine installation.  PF\_IsRenderEngine(  PF\_Boolean \*render\_enginePB);  As of AE6.5, this function returns TRUE if the installation is the render engine, or if the After Effects is being run with no UI, or if After Effects is in watched folder mode. |
| PF\_AppColorPickerDialog | Displays the After Effects color picker dialog (which may be the  system color picker, depending on the user’s preferences). Will return PF\_Interrupt\_CANCEL if user cancels dialog. Returned color is in the project’s working color space.  PF\_AppColorPickerDialog(  const A\_char \*dialog\_titleZ0, const PF\_PixelFloat \*sample\_colorP, PF\_PixelFloat \*result\_colorP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetMouse | Returns the position of the mouse in the custom UI coordinate space.  PF\_GetMouse(  PF\_Point \*pointP); |
| PF\_InvalidateRect | Queue up a [*redraw*](#_bookmark480) of a specific area of the custom UI for an effect. Only valid while handling a non-drawing event in the effect. Specify rectP0 as NULL to invalidate the entire window. The redraw will happen at the next available idle moment after returning from the event. Set the PF\_EO\_UPDATE\_NOW event outflag to update the window immediately after the event returns.  PF\_InvalidateRect(  const PF\_ContextH contextH, const PF\_Rect\* rectP0); |
| PF\_ConvertLocalToGlobal | Converts from the custom UI coordinate system to global screen coordinates. Use only during custom UI event handling.  PF\_ConvertLocalToGlobal(  const PF\_Point \*localP,  PF\_Point \*globalP); |

**ADVANCED APPSUITE: YOU CAN DO THAT?!**

PF\_AdvAppSuite was originally designed for some pretty nefarious purposes; an external

application was pretending to be an After Effects plug-in, and required ways to notify After Effects of the changes it had made to the project. Our API impurity is your gain.

**TABLE 33: AE\_ADVAPPSUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_SetProjectDirty | Tells After Effects that the project has been changed since it was last saved.  PF\_SetProjectDirty(void); |
| PF\_SaveProject | Saves the project to the current path. To save the project elsewhere, use  [AEGP\_SaveProjectToPath()](#_bookmark564).  PF\_SaveProject(void); |
| PF\_SaveBackgroundState | Stores the background state (After Effects’ position in the stacking order of open applications and windows).  PF\_SaveBackgroundState(void); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_ForceForeground | Brings After Effects to the front of all currently open applications and windows.  PF\_ForceForeground(void); |
| PF\_RestoreBackgroundState | Puts After Effects back where it was, in relation to other applications and windows.  PF\_RestoreBackgroundState(void); |
| PF\_RefreshAllWindows | Forces all After Effects windows to update. Note that although the Composition panel will be refreshed, this does not guarantee a new frame will be sent to External Monitor Preview plug-ins.  PF\_RefreshAllWindows(void); |
| PF\_InfoDrawText | Writes text into the After Effects info palette.  PF\_InfoDrawText(  const A\_char \*line1Z0,  const A\_char \*line2Z0); |
| PF\_InfoDrawColor | Draws the specified color in the After Effects info palette (alpha is ignored).  PF\_InfoDrawColor(  PF\_Pixel color); |
| PF\_InfoDrawText3 | Writes three lines of text into the After Effects info palette.  PF\_InfoDrawText3(  const A\_char \*line1Z0,  const A\_char \*line2Z0,  const A\_char \*line3Z0); |
| PF\_InfoDrawText3Plus | Writes three lines of text into the After Effects info palette, with portions of the second and third lines left and right justified.  PF\_InfoDrawText3Plus(  const A\_char \*line1Z0,  const A\_char \*line2\_jrZ0,  const A\_char \*line2\_jlZ0,  const A\_char \*line3\_jrZ0,  const A\_char \*line3\_jlZ0); |
| PF\_AppendInfoText | Appends characters to the currently-displayed info text.  PF\_AppendInfoText(  const A\_char \*appendZ0); |

PF\_AdvTimeSuite provides several functions to match how After Effects displays time. In fact, these are the same functions we use internally.

**TABLE 34: PF\_ADVTIMESUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_FormatTimeActiveItem | Given a time value and scale, returns a formatted string representing that time. If durationB is TRUE, appropriate units will be appended.  PF\_FormatTimeActiveItem(  A\_long time\_valueUL,  A\_u\_long time\_scaleL,  PF\_Boolean durationB,  A\_char \*time\_buf); |
| PF\_FormatTime | Contextualizes the formatted time string for the given PF\_InData  and PF\_EffectWorld (i.e., layer time).  PF\_FormatTime(  PF\_InData \*in\_data,  PF\_EffectWorld \*world,  A\_long time\_valueUL,  A\_u\_long time\_scaleL,  PF\_Boolean durationB,  A\_char \*time\_buf); |

**TABLE 34: PF\_ADVTIMESUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_FormatTimePlus | Allows you to select composition or layer time.  PF\_FormatTimePlus(  PF\_InData \*in\_data,  PF\_EffectWorld \*world,  A\_long time\_valueUL,  A\_u\_long time\_scaleL,  PF\_Boolean comp\_timeB,  PF\_Boolean durationB,  A\_char \*time\_buf); |
| PF\_GetTimeDisplayPref | Returns the starting frame number (specified by the user in  composition settings), and the composition’s time display preferences.  PF\_GetTimeDisplayPref( PF\_TimeDisplayPref2 \*tdp,  A\_long \*starting\_num);  typedef struct {  A\_char display\_mode;  A\_char framemax;  A\_char frames\_per\_foot;  A\_char frames\_start;  A\_Boolean nondrop30B;  A\_Boolean honor\_source\_timecodeB; A\_Boolean use\_feet\_framesB;  } PF\_TimeDisplayPrefVersion2; |

**AFFECTING THE TIMELINE**

Long ago, we helped a developer integrate their stand-alone tracker with After Effects by

exposing a set of functions to give them some way to notify us of, and be notified of, changes

to the timeline. With the numerous AEGP API calls available, these aren’t used much, but they’re still available. Don’t confuse this suite with [AEGP\_ItemSuite](#_bookmark567).

**TABLE 35: PF\_ADVITEMSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_MoveTimeStep | Moves current time num\_stepsL in the specified direction.  PF\_MoveTimeStep(  PF\_InData \*in\_data,  PF\_EffectWorld \*world,  PF\_Step time\_dir,  A\_long num\_stepsL); |
| PF\_MoveTimeStep ActiveItem | Moves num\_stepsL in the specified direction, for the active item.  PF\_MoveTimeStepActiveItem( PF\_Step time\_dir,  A\_long num\_stepsL); |
| PF\_TouchActiveItem | Tells After Effects that the active item must be updated.  PF\_TouchActiveItem (void); |
| PF\_ForceRerender | Forces After Effects to rerender the current frame.  PF\_ForceRerender(  PF\_InData \*in\_data,  PF\_EffectWorld \*world); |
| PF\_EffectIsActive OrEnabled | Returns whether the effect which owns the PF\_ContextH is  currently active or enabled (if it isn’t, After Effects won’t be listening for function calls from it).  PF\_EffectIsActiveOrEnabled( PF\_ContextH contextH,  PF\_Boolean \*enabledPB); |

Some file types contain more than just pixel data; use [PF\_ChannelSuite](#_bookmark360) to determine

whether such information is present, and the macros in AE\_ChannelSuites.h to retrieve it in the format you need.

**TABLE 36: PF\_CHANNELSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetLayerChannelCount | Retrieves the number of auxiliary channels associated with the indexed layer.  PF\_GetLayerChannelCount(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index,  A\_long \*num\_channelsPL); |
| PF\_GetLayerChannel IndexedRefAndDesc | Retrieves (by index) a reference to, and description of, the specified channel.  PF\_GetLayerChannelIndexedRefAndDesc( PF\_ProgPtr effect\_ref  PF\_ParamIndex param\_index PF\_ChannelIndex channel\_index, PF\_Boolean \*foundPB,  PF\_ChannelRef \*channel\_refP, PF\_ChannelDesc \*channel\_descP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetLayerChannel TypedRefAndDesc | Retrieves an auxiliary channel by type. Returned information is valid only if  foundPB returns TRUE.  PF\_GetLayerChannelTypedRefAndDesc( PF\_ProgPtr effect\_ref,  PF\_ParamIndex param\_index, PF\_ChannelType channel\_type, PF\_Boolean \*foundPB,  PF\_ChannelRef \*channel\_refP, PF\_ChannelDesc \*channel\_descP);  PF\_DataType will be one of the following:  PF\_DataType\_FLOAT 34 bytes  PF\_DataType\_DOUBLE 38 bytes  PF\_DataType\_LONG 34 bytes  PF\_DataType\_SHORT 32 bytes  PF\_DataType\_FIXED\_16\_16 34 bytes  PF\_DataType\_CHAR 31 byte  PF\_DataType\_U\_BYTE 31 byte  PF\_DataType\_U\_SHORT 32 bytes  PF\_DataType\_U\_FIXED\_16\_16 34 bytes  PF\_DataType\_RGB 3 bytes  PF\_ChannelType will be one of the following:  PF\_ChannelType\_DEPTH PF\_ChannelType\_NORMALS PF\_ChannelType\_OBJECTID PF\_ChannelType\_MOTIONVECTOR PF\_ChannelType\_BK\_COLOR PF\_ChannelType\_TEXTURE PF\_ChannelType\_COVERAGE PF\_ChannelType\_NODE PF\_ChannelType\_MATERIAL PF\_ChannelType\_UNCLAMPED PF\_ChannelType\_UNKNOWN |
| PF\_CheckoutLayerChannel | Retrieves the PF\_ChannelChunk containing the data associated with the given PF\_ChannelRefPtr.  PF\_CheckoutLayerChannel(  PF\_ProgPtr effect\_ref, PF\_ChannelRefPtr channel\_refP, long what\_time,  long duration,  unsigned long time\_scale,  PF\_DataType data\_type, PF\_ChannelChunk \*channel\_chunkP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_CheckinLayerChannel | Checks in the PF\_ChannelChunk. Always, always, always check the data back in.  PF\_CheckinLayerChannel(  PF\_ProgPtr effect\_ref, PF\_ChannelRefPtr channel\_refP, PF\_ChannelChunk \*channel\_chunkP); |

### MOTION BLUR

Effects handle their own motion blur, using PF\_InData>[shutter\_angle](#_bookmark117) along with PF\_InData>[shutter\_phase](#_bookmark123). The plug-in must set [PF\_OutFlag\_I\_USE\_SHUTTER\_ANGLE](#_bookmark159) so After Effects knows it needs this information.

They must [*check out*](#_bookmark289) their own parameters at other times to examine their change over the shutter interval. If the plug-in checks out parameters outside this interval, set [PF\_OutFlag\_WIDE\_TIME\_INPUT](#_bookmark146). Doing so allows After Effects to compare the

parameters within the sampling interval, and determine if they’ve changed.

## WORKING WITH PATHS

**ACCESSING PATH DATA**

Paths differ from other parameter types, in that their values are not directly accessible. In addition to checking them out and in (like layer parameters), you must use our path data

function suites to obtain the details of the path at a given time. See [PF\_PathQuerySuite](#_bookmark371)

and [PF\_PathDataSuite](#_bookmark368). Never use the values present in a path parameter when it’s

passed to you, without first checking it out; while deleted paths will not be available, further updating is done “lazily” (later); your effect won’t see these changes unless it checks out the path.

**MANIPULATING PATH DATA**

You can also use the [AEGP\_MaskOutlineSuite](#_bookmark658) to manipulate paths. See “[cheating](#_bookmark718)”.Path parameters are treated as opaque blobs of data; get and set functions must be used to access

and manipulate them. Like layer parameters, they must be checked out (and in!) by effects which access them.

**VERTICES**

Path vertices are more complex than simple points. All member variables are PF\_FpLongs

(doubles), and are in the layer’s coordinate space.

**TABLE 37: PF\_PATHVERTEX**

|  |  |
| --- | --- |
| **Member** | **Description** |
| x | The location of the vertex. |
| y |
| tan\_in\_x | The incoming tangent point. |
| tan\_in\_y |
| tan\_out\_x | The outgoing tangent point. |
| tan\_out\_y |

**PF\_PATHDATASUITE**

This suite provides information about paths (sequences of vertices).

**TABLE 38: PF\_PATHDATASUITE1**

|  |  |
| --- | --- |
| **Function** | **Description** |
| PF\_PathIsOpen | Returns TRUE if the path is not closed (if the beginning and end vertex are not identical).  PF\_PathIsOpen(  PF\_ProgPtr effect\_ref0,  PF\_PathOutlinePtr pathP,  PF\_Boolean \*openPB); |
| PF\_PathNumSegments | Retrieves the number of segments in the path. N segments means there are segments [0.N-1]; segment J is defined by vertex J and J+1.  PF\_PathNumSegments(  PF\_ProgPtr effect\_ref0,  PF\_PathOutlinePtr pathP,  A\_long \*num\_segmentsPL); |

|  |  |
| --- | --- |
| **Function** | **Description** |
| PF\_PathVertexInfo | Retrieves the PF\_PathVertex for the specified path. The range of  points is [0.num\_segments]; for closed paths, vertex[0] == vertex[num\_segments].  PF\_PathVertexInfo(  PF\_ProgPtr effect\_ref0,  PF\_PathOutlinePtr pathP,  A\_long which\_pointL,  PF\_PathVertex \*vertexP); |
| PF\_PathPrepare SegLength | This fairly counter-intuitive function informs After Effects that you’re going to ask for the length of a segment (using PF\_PathGetSegLength below), and it’d better get ready. frequencyL indicates how many times you’d like us to sample the  length; our internal effects use 100.  PF\_PathPrepareSegLength(  PF\_ProgPtr effect\_ref0,  PF\_PathOutlinePtr pathP,  A\_long which\_segL,  A\_long frequencyL,  PF\_PathSegPrepPtr \*lengthPrepPP); |
| PF\_PathGetSegLength | Retrieves the length of the given segment.  PF\_PathGetSegLength(  PF\_ProgPtr effect\_ref0, PF\_PathOutlinePtr pathP, A\_long which\_segL,  PF\_PathSegPrepPtr \*lengthPrepP0, PF\_FpLong \*lengthPF); |
| PF\_PathEvalSegLength | Retrieves the location of a point lengthF along the given path segment.  PF\_PathEvalSegLength(  PF\_ProgPtr effect\_ref0, PF\_PathOutlinePtr pathP, PF\_PathSegPrepPtr \*lengthPrepPP0, A\_long which\_segL,  PF\_FpLong lengthF,  PF\_FpLong \*x,  PF\_FpLong \*y); |

|  |  |
| --- | --- |
| **Function** | **Description** |
| PF\_PathEvalSegLength Deriv1 | Retrieves the location, and the first derivative, of a point lengthF along the given path segment. If you’re not sure why you’d ever need this, don’t use it. Math is hard.  PF\_PathEvalSegLengthDeriv1(  PF\_ProgPtr effect\_ref0, PF\_PathOutlinePtr pathP, PF\_PathSegPrepPtr \*lengthPrepPP0, A\_long which\_segL,  PF\_FpLong lengthF,  PF\_FpLong \*x,  PF\_FpLong \*y,  PF\_FpLong \*deriv1x,  PF\_FpLong \*deriv1y); |
| PF\_PathCleanup SegLength | Call this when you’re finished evaluating that segment length, so After Effects can properly clean up the PF\_PathSegPrepPtr.  PF\_PathCleanupSegLength(  PF\_ProgPtr effect\_ref0, PF\_PathOutlinePtr pathP, A\_long which\_segL,  PF\_PathSegPrepPtr \*lengthPrepPP); |
| PF\_PathIsInverted | Returns TRUE if the path is inverted.  PF\_PathIsInverted(  PF\_ProgPtr effect\_ref,  PF\_PathID unique\_id,  PF\_Boolean \*invertedB); |

|  |  |
| --- | --- |
| **Function** | **Description** |
| PF\_PathGetMaskMode | Retrieves the mode for the given path.  PF\_PathGetMaskMode(  PF\_ProgPtr effect\_ref,  PF\_PathID unique\_id,  PF\_MaskMode \*modeP);  Mask mode is one of the following:  PF\_MaskMode\_NONE PF\_MaskMode\_ADD PF\_MaskMode\_SUBTRACT PF\_MaskMode\_INTERSECT PF\_MaskMode\_LIGHTEN PF\_MaskMode\_DARKEN PF\_MaskMode\_DIFFERENCE PF\_MaskMode\_ACCUM |
| PF\_PathGetName | Retrieves the name of the path (up to PF\_MAX\_PATH\_NAME\_LEN  long).  PF\_PathGetName(  PF\_ProgPtr effect\_ref,  PF\_PathID unique\_id,  A\_char \*nameZ); |

**PF\_PATHQUERYSUITE**

This suite is used to identify and access the paths associated with the effect’s source layer.

**TABLE 39: PF\_PATHQUERYSUITE**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_NumPaths | Retrieves the number of paths associated with the effect’s source layer.  PF\_NumPaths(  PF\_ProgPtr effect\_ref,  A\_long \*num\_pathsPL); |
| PF\_PathInfo | Retrieves the PF\_PathID for the specified path.  PF\_PathInfo(  PF\_ProgPtr effect\_ref,  A\_long indexL,  PF\_PathID \*unique\_idP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_CheckoutPath | Acquires the PF\_PathOutlinePtr for the path at the specified time.  PF\_CheckoutPath(  PF\_ProgPtr effect\_ref,  PF\_PathID unique\_id,  A\_long what\_time,  A\_long time\_step,  A\_u\_long time\_scale, PF\_PathOutlinePtr \*pathPP); |
| PF\_CheckinPath | Releases the path back to After Effects. Always do this, regardless of any error conditions encountered. Every checkout must be balanced by a checkin, or pain will ensue.  PF\_CheckinPath(  PF\_ProgPtr effect\_ref,  PF\_PathID unique\_id,  PF\_Boolean changedB, PF\_PathOutlinePtr pathP); |

**ACCESSING CAMERA AND LIGHT INFORMATION**

Using functions provided in the [AEGP\_PFInterfaceSuite](#_bookmark709), effects can access camera and lighting information for the layer to which they’re applied; see the Resizer sample. You can also use many of the other functions from AE\_GeneralPlug.h; the possibilities are vast.

### COLOR SPACE CONVERSION

Different pixel formats are useful for different operations. After Effects exposes its internal functions through PF\_ColorCallbacksSuite. Here are the supported formats.

**TABLE 40: PIXEL TYPES FOR DIFFERENT COLOR SPACES**

|  |  |
| --- | --- |
| **Pixel Type** | **Data Structure** |
| 8 bpc ARGB | typedef struct {  A\_u\_char alpha, red, green, blue;  } PF\_Pixel8; |
| 16 bpc ARGB | typedef struct {  A\_u\_short alpha, red, green, blue;  } PF\_Pixel16; |
| 32 bpc ARGB | typedef struct {  PF\_FpShort alpha, red, green, blue;  } PF\_PixelFloat, PF\_Pixel32; |
| HLS (Hue, Lightness, Saturation) | typedef PF\_Fixed PF\_HLS\_PIXEL[3] |
| YIQ (luminance, in-phase chrominance, quadrature chrominance) | typedef PF\_Fixed PF\_YIQ\_PIXEL[3] |

Plug-ins can draw on image processing algorithms written for nearly any color space by using the following callback functions.

**TABLE 41: COLOR SPACE CONVERSION CALLBACKS**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| RGBtoHLS | Given an RGB pixel, returns an HLS (hue, lightness, saturation) pixel. HLS values are scaled from 0 to 1 in fixed point.  Replaces PF\_RGB\_TO\_HLS. |
| HLStoRGB | Given an HLS pixel,returns an RGB pixel.  Replaces PF\_HLS\_TO\_RGB. |
| RGBtoYIQ | Given an RGB pixel, returns a YIQ (luminance, inphase chrominance, quadrature chrominance) pixel. Y is 0 to 1 in fixed point, I is -0.5959 to 0.5959 in fixed point, and Q is -0.5227 to 0.5227 in fixed point.  Replaces PF\_RGB\_TO\_YIQ. |
| YIQtoRGB | Given a YIQ pixel, returns an RGB pixel.  Replaces PF\_YIQ\_TO\_RGB. |

**TABLE 41: COLOR SPACE CONVERSION CALLBACKS**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| Luminance | Given an RGB pixel, returns 100 times its luminance value (0 to 25500).  Replaces PF\_LUMINANCE. |
| Hue | Given an RGB pixel, eturns its hue angle mapped from 0 to 255, where 0 is 0 degrees and 255 is 360 degrees.  Replaces PF\_HUE. |
| Lightness | Given an RGB pixel, returns its lightness value (0 to 255).  Replaces PF\_LIGHTNESS. |
| Saturation | Given an RGB pixel, returns its saturation value (0 to 255).  Replaces PF\_SATURATION. |

## CHANGING PARAMETER ORDERS, THE NICE WAY

It is possible to add or remove parameters from a plug-in, without forcing users to re-apply all instances of that plug-in to use the updated version. However, some advance planning on your part is necessary to allow for such changes. Your users (and technical support staff) will appreciate the effort.

You must first create a parameter array index. During *PF\_Cmd\_PARAM\_SETUP*, assign index values to each parameter as you add them, using a simple enumeration. The order of enumeration corresponds to the order in which the parameters are registered during

*PF\_Cmd\_PARAM\_SETUP*, which in turn determines the order in which they appear in the Effect Control and Timeline panels.

Create another enumeration for disk IDs. The order of this enumeration must *not* be

changed, though you may add to the end of this list. Note that the order of this list need not correspond with that of the parameter array index. Parameter disk IDs should range from 1 to 9999. Why not zero? Long story...

*In the early “wild west” days of After Effects plug-in programming, it was fairly common for*

*developers not to bother with setting IDs. After Effects, realizing this, checked the ID of the first parameter added by that effect; if it was zero, it was assumed that the programmer hadn’t*

*bothered to ID params; After Effects then assigned each its own ID. This assumption works fine if you never set param IDs, but not so well if you start numbering your IDs from NULL. That’s*

*why.*

Before calling PF\_ADD\_PARAM(), specify the disk ID in the PF\_ParamDef.uu.id field. If no

value is specified, After Effects makes parameters sequential starting with 1. The parameter’s information is tagged with this ID when saved. In this way, After Effects can still

understand that, although your “Foobarocity” slider is now the fourth parameter passed, it’s the same parameter as when it was second.

To delete a parameter without forcing re-application, remove the code which creates it and its entry in the parameter array index list. However, *do not* remove its entry in the disk ID

list. To add a new parameter, add an entry in the appropriate location in the parameter array indices list, add the parameter creation code, and append the disk ID to the end of the disk

ID enumeration. To re-order, change the parameter array index list and reorder the parameter creation code appropriately.

**CHANGE DEFAULTS? CHANGE IDS**

If you don’t, if someone saves a project with the old default and then reads it in with the new effect installed, that parameter will change to the new default value. Presto! Instant support call. This is another prime use case for [PF\_ParamFlag\_USE\_VALUE\_FOR\_OLD\_PROJECTS](#_bookmark222).

**TIPS AND TRICKS**

**BEST PRACTICES**

If your prototypes are anything like ours, the first version of your plug-in that runs without crashing differs radically from the version that actually ships. How your plug-in responds to things like downsampling, errors and exceptions, pixel aspect ratio, out-of-memory

situations, and being interrupted while processing determines how usable it is (and how many support requests you’ll have to handle).

**RESPONSIVENESS**

Make your plug-ins as responsive as possible using [PF\_ABORT()](#_bookmark285) and [PF\_PROGRESS()](#_bookmark287). We actually test all our effects for interruptability; you’d be surprised how cranky users can get waiting for your pokey effect to finish processing a film resolution sequence! After Effects’

iteration functions inherently provide this functionality; you don’t need to worry about calling the above functions from within your pixel processing functions.

**MAKE YOUR EFFECT EASY TO FIND**

It’s possible to have your effect show up in the “Effects & Presets” palette when users search for something other than the plug-in’s name. Apply your effect (leaving the settings at

default, unless you’re very certain the user will want something different when they search for the given term), and select “Save selection as animation preset” from the effect controls

palette. Save it to the name by which you want users to find the plug-in. Have your plug-in’s installer put the resultant .ffx file into the \Presets directory, next to the After Effects

executable. Your preset will show up when users search for the name to which it was saved.

**SAMPLING PIXELS AT (X,Y)**

Sometimes, instead of just processing every pixel, you’ll want to get to a specific offset within the input frame. Below is one way to sample the pixel at a given (x,y) location; similar code

could be used to write to the given location.

PF\_Pixel \*sampleIntegral32(PF\_EffectWorld &def, int x, int y){ return (PF\_Pixel\*)((char\*)def.data +

(y \* def.rowbytes) +

(x \* sizeof(PF\_Pixel)));

}

PF\_Pixel16 \*sampleIntegral64(PF\_EffectWorld &def, int x, int y){ assert(PF\_WORLD\_IS\_DEEP(&def));

return (PF\_Pixel16\*)((char\*)def.data +

(y \* def.rowbytes) +

(x \* sizeof(PF\_Pixel16)));

}

Special thanks to Paul Miller of Profound Effects, who answered that question on the

ae\_api mailing list.

**WHERE’S THE CENTER OF A PIXEL?**

Deeeeeep, man. After Effects rotates around the upper left corner of the upper left pixel when the anchor point (see User Documentation) is (0,0). However, the subpixel sample and area sample callbacks actually treat (.0, .0) as a direct hit. To compensate for this, subtract 0.5 from x and y values before calling those functions. The matrix functions ([transform\_world](#_bookmark268)) don't have this problem.

When translating an image by a subpixel amount, make the output layer one pixel wider than its input, and leave the origin at (0,0).

**CLEAN SLATE**

You don’t necessarily begin effect processing with a clean output slate. Our Gaussian blur filter, in an effort to do so, performs the following before rendering:

src\_rect.left = in\_data>output\_origin\_x; src\_rect.right = src\_rect.left + input>width; src\_rect.top = in\_data>output\_origin\_y; src\_rect.bottom = src\_rect.top + input>height; err = PF\_FILL(NULL, NULL, output);

if (!err) {

err = PF\_COPY(&params[0]>u.ld, output, NULL, &src\_rect);

}

**CACHING BEHAVIOR**

After Effects provides numerous ways to specify caching behavior. [PF\_OutFlag\_NON\_PARAM\_VARY](#_bookmark147), [PF\_OutFlag\_WIDE\_TIME\_INPUT](#_bookmark146), [PF\_OutFlag\_I\_USE\_SHUTTER\_ANGLE](#_bookmark159), [PF\_OutFlag\_I\_SYNTHESIZE\_AUDIO](#_bookmark163), [PF\_OutFlag2\_I\_USE\_3D\_CAMERA](#_bookmark170), and [PF\_OutFlag2\_I\_USE\_3D\_LIGHTS](#_bookmark172) all influence caching decisions.

Supporting [dynamic outflags](#_bookmark167) can greatly improve performance, preventing After Effects from invalidating your effect's cache as aggressively as it otherwise would.

Confirm that your plug-in performs well with different After Effects cache settings. Does

your plug-in get called to update as often as expected, or does After Effects think it has valid pixels when you think it doesn’t?

**GLOBAL PERFORMANCE CACHE CONSIDERATONS**

With the new caching in CS6, you may need to clear cached frames after changing your

effect’s rendering, so that frames rendered and stored in the cache prior to the change will not be reused. To do so manually during development:

1. In Preferences > Media & Disk Cache, disable the Disk Cache
2. Click "Empty Disk Cache" just to be sure (disabling the Disk Cache in step 1 only disables the \*writing\* of disk cache, not necessarily the usage)
3. Relaunch

If you ever encounter a glitch, it likely a legitimate bug in your effect, such as improper rectangle handling in SmartFX.

On the other hand, if you fix a rendering bug in your plug-in and ship an update, you can’t

expect all users will empty their disk caches. A user may have a disk cache of the buggy frame and it needs to be invalidated. What to do? Update your plug-in's effect version. This value (and the AE build number) is part of the cache key, so if you update it any frames cached

containing content from your plug-in will no longer match.

**SOME THOUGHTS ON TIME FROM A LONG-TIME DEVELOPER**

Stoney Ballard put together the following summary of how time works with effects; you may find it helpful.

There are five in\_data parameters that describe time to a filter:

current\_time time\_step local\_time\_step total\_time time\_scale

Their values are dependent on: The frame being rendered

The duration of the layer and composition

The frame rate of the comp Any Time Stretch

Any Time Remapping

The time behavior of an outer composition (one enclosing the composition with the layer being filtered)

The setting of the "Preserve frame rate when nested or in render queue" (PFR) switch

The frame being rendered affects current\_time. It is expressed in the local (layer) time

system. If the PFR switch is off, current\_time may be any non-negative value. If on, it will be restricted to a multiple of time\_step and local\_time\_step. Layer duration affects only total\_time. Comp duration is a factor only when Time Remapping (TR) is on. In that case, total\_time is the larger of layer duration and composition duration. Composition frame rate affects only the time\_scale. Time Stretch affects only time\_step and local\_time\_step. If the time stretch is negative, these values are negative. Even if the layer's duration (as seen in the comp) changes, total\_time remains unaffected. This works as if Time Stretch was *above* a filter, but *below* an outer comp. PFR does not alter the effect of Time Stretch. Time

Stretch is different than an outer comp, since it affects both step params equally, while an outer comp affects only time\_step.

Time Remapping happens *below* the filter, so that it does not affect the time params other than the total\_time. When TR is on, the layer is lengthened to the same as the comp (but never shortened), regardless of how much time it actually takes, or where in the comp the

layer is. This may cause total\_time to be larger. It has nothing to do with the actual time map, just whether or not it's enabled.

The biggest variation comes from being nested in an outer comp, unless PFR is on. When PFR is on, a filter is completely isolated from time variations in an outer comp. Of course,

current\_time will not necessarily move in increments of time\_step in that case. It may skip frames or go backwards.

When PFR is off, local\_time\_step, total\_time, and time\_scale remain set to what they were for the inner comp, but time\_step contains the time to the next frame in the outer

comp, expressed in the local time system. This may be any value, including 0. This can be interpreted as an instantaneous time rate, rather than a duration. A 0 value can last for an

arbitrary number of rendered frames, but the current\_time won't change on the local layer. Looked at from the other direction:

current\_time is quantized to time\_step intervals unless rendering an outer comp with PFR off for the inner comp. This is the current time in the layer, not in any comp.

The value of local\_time\_step is affected only by Time Stretch. It can never be zero, but it can be negative.

time\_step and local\_time\_step are always the same value unless rendering an outer comp with PFR off. time\_step is also affected by the time behavior of an outer comp (with PFR

off). It can have any value, positive, negative, or zero, and can be different for every frame (of the outer comp). time\_step can be used to determine the duration of the current frame

(with PFR off).

total\_time is the duration of the layer, unless Time Remapping is on, which makes it the larger of the layer duration and the duration of the comp.

time\_scale is the scale such that total\_time / time\_scale is the layer duration in seconds in its comp. It is affected only by the comp frame rate, although presumably all the time

values could be scaled proportionately for any reason.

A layer's intrinsic frame rate (if it has one) is not visible anywhere, although it's usually the

same as the comp frame rate. If a filter needs to access the actual frames of a clip, it can do so only by being in a comp of the same frame rate, and with no Time Stretch or Time

Remapping applied to its layer. It should use local\_time\_step to determine where the frames are.

**RATE X TIME == PAIN**

Be careful if one of your parameters is a speed or velocity parameter. Consider the ripple

effect. It assumes a constant and uses the current time to determine how far along the ripple has gone (d = v \* t). If the user interpolates the speed over time, you should integrate the velocity function from time zero to the current time. Ripple does *not* do this, but provides a “phase” parameter that the user can interpolate as they wish, providing correct results as long as the speed is set to zero. If you want to provide the correct behavior, you can sample (and

integrate) the speed parameter from the beginning of time until the current time using PF\_CHECKOUT\_PARAM(), or you can provide a “phase” or “distance” parameter and warn the user about interpolating the speed. The cost of checking out many parameter values is

negligible compared to rendering, and is the recommended approach.

If you check out parameter values at other times, or use layer parameters at all, you *must*

check in those parameters when finished, even if an error has occurred. Remember, checked- out parameters are read-only.

**TESTING**

Try using your plug-in in RAM previews to ensure you handle out-of-memory conditions gracefully. Does your plug-in handle running out of memory gracefully? If you receive [PF\_Err\_OUT\_OF\_MEMORY](#_bookmark248) when requesting memory, do you pass it back to After Effects?

What happens when your video effect is applied to an audio-only layer? Test with projects created using older versions of your plug-in.

4: SMARTFX

The SmartFX API provides bidirectional communication between effects and After Effects, enabling many performance optimizations and providing previously unavailable dependency information. This extension of the effect API is the way to implement 32-bit per channel

support in After Effects.

Normal effect plug-ins are given a full-sized input buffer, and asked to render a full-sized output buffer. While output [extent\_hint](#_bookmark119) specifies the portion of the output buffer that

must actually be filled, this scheme is still very inefficient if the effect does not need its entire input. Also, many effects don’t use extent hints.

**THE WAY THINGS WERE**

Consider a blur effect applied to a huge layer which is mostly off-screen, or viewed through a small region of interest, or masked down to a small size. Only a small section of the output needs to be rendered, indicated to the effect using the output extent\_hint. Only a small

section of the input to be blurred is needed as well - the output extent\_hint expanded by the blur radius. However, using the legacy effects API, there is no way for After Effects to know this, so the entire layer is passed to the plug-in. These extra pixels can be extremely

expensive and wasteful to compute, especially in the case of prior effects or nested comps.

**THE WAY THINGS ARE NOW**

SmartFX solves this problem by reversing the calling sequence. The effect is told how much of its output is required, and must explicitly *ask* the host for the inputs it needs. The render process is split into two parts: pre-render and render.

During pre-render, the effect describes the input pixel data it needs; this necessary input can vary based on anything you like (non-input layer parameters, non-layer parameters,

information from in\_data, settings in sequence data...). The effect must also return the extent of the resulting output, which may be smaller than the requested size if there are empty pixels in the requested portion of the layer.

During the render stage, the effect can *only* retrieve pixels that it has previously requested.

This two-pass approach facilitates many important optimizations. For example, an effect

which multiplies or mattes one input against another might discover that its first input is not

needed at all, if the mask does not intersect it. There are also important optimizations that are performed internally by After Effects to ensure that image buffers are copied as little as possible, and these optimizations are only possible after the host knows the buffer sizes and for all inputs and outputs.

Like AEGPs, SmartFX plug-ins are never unloaded by After Effects.

**CONTENT BOUNDS**

The content bounds of a node are the largest possible result rectangle that can be returned from a call to PreRender. It absolutely cannot vary depending on current render request or anything else. It should be calculated carefully, not loosely.

This calculation is very important. It is an intrinsic property of the node (and its inputs) and is fixed once the graph is built. Violation of it can and probably will cause all sorts of

problems in various pieces of code.

**HOW TO SMARTIFY**

Effects which set [PF\_OutFlag2\_SUPPORTS\_SMART\_RENDER](#_bookmark177) will receive the SmartFX

calls [*PF\_Cmd\_SMART\_PRE\_RENDER*](#_bookmark96) and [*PF\_Cmd\_SMART\_RENDER*](#_bookmark98), instead of the older [*PF\_Cmd\_FRAME\_SETUP*](#_bookmark87) / [*PF\_Cmd\_RENDER*](#_bookmark89)/[*PF\_Cmd\_FRAME\_SETDOWN*](#_bookmark91) sequence. To preserve compatibility with non-smartified hosts, you may want to continue

supporting the older commands too.

### PF\_CMD\_SMART\_PRE\_RENDER

After Effects requests output from the effect. The effect tells After Effects what input it needs to generate that output, through the use of callback functions, and by manipulating

the structures in the extra parameter. An effect cannot access the pixels of any layer inputs it has not checked out during *PF\_Cmd\_SMART\_PRE\_RENDER*. So all layer inputs that

an effect might possibly need must be checked out in advance using checkout\_layer. If an effect might need certain layer inputs, they must be checked out now, even if later during

rendering the effect may decide that the layer isn’t needed. Also, since no parameter array is passed to SmartFX during *PF\_Cmd\_SMART\_PRE\_RENDER* or

[*PF\_Cmd\_SMART\_RENDER*](#_bookmark415), any non-layer parameters needed must be retrieved using

[PF\_CHECKOUT\_PARAM](#_bookmark288)

**TABLE 42: PF\_PRERENDEREXTRA**

|  |  |
| --- | --- |
| Member | Purpose |
| PF\_PreRenderInput | Describes what After Effects needs rendered (in the PF\_RenderRequest), and the bit depth requested (in the aptly-named bitdepth member).  typedef struct {  PF\_LRect rect;  PF\_Field field; PF\_ChannelMask channel\_mask;  PF\_Boolean preserve\_rgb\_of\_zero\_alpha; char unused[3];  long reserved[4];  }PF\_RenderRequest;  rect is in layer coordinates. field is also relative to the layer origin; whether the active field falls on even or odd scanlines of the output buffer depends on the origin of the output buffer.  channel\_mask specifies for which channels the effect should provide output. Data written to other channels will not be honored. It will be one or more of the following, or’d together:  PF\_ChannelMask\_ALPHA PF\_ChannelMask\_RED PF\_ChannelMask\_GREEN PF\_ChannelMask\_BLUE PF\_ChannelMask\_ARGB  If preserve\_rgb\_of\_zero\_alpha pixels is TRUE,the effect must propagate the color content of transparent pixels through to the output. This is related to, but distinct from, [PF\_OutFlag2\_REVEALS\_ZERO\_ALPHA](#_bookmark176), which tells After Effects that the effect may set alpha to non-zero values for such pixels, restoring them to visibility. |

|  |  |
| --- | --- |
| Member | Purpose |
| PF\_PreRenderOutput | Filled in by the effect to tell After Effects what output it plans to generate, based on the input.  typedef struct {  PF\_LRect result\_rect;  PF\_LRect max\_result\_rect;  PF\_Boolean solid;  PF\_Boolean reserved; PF\_RenderOutputFlags flags;  void\* pre\_render\_data; PF\_DeletePreRenderDataFunc func;  } PF\_PreRenderOutput;  pre\_render\_data will be passed back to the effect during  [*PF\_Cmd\_SMART\_RENDER*](#_bookmark98).  Currently, the only PF\_RenderOutputFlags is  PF\_RenderOutputFlag\_RETURNS\_EXTRA\_PIXELS. |

|  |  |
| --- | --- |
| Member | Purpose |
| PF\_PreRenderCallbacks | Currently, there is only one callback - checkout\_layer. checkout\_idL is chosen by the effect. It must be positive and unique. After Effects populates the PF\_CheckoutResult.  PF\_Err checkout\_layer(  PF\_ProgPtr effect\_ref,  PF\_ParamIndex index,  A\_long checkout\_idL, const PF\_RenderRequest \*req,  A\_long what\_time,  A\_long time\_step,  A\_u\_long time\_scale,  PF\_CheckoutResult \*result);  typedef struct {  PF\_LRect result\_rect;  PF\_LRect max\_result\_rect;  PF\_RationalScale par;  long solid;  PF\_Boolean reservedB[3];  A\_long ref\_width;  A\_long ref\_height;  } PF\_CheckoutResult;  result\_rect can be empty. max\_result\_rect is the largest the output could possibly be, if the host asked for all of it. If solid is TRUE, the entire result\_rect has opaque alpha.  ref\_width and ref\_height are the original dimensions of the layer, before any effects are applied, disregarding any downsample factors. This will be the size of the composition for collapsed layers.  There is a bug in 11.0 with the Global Performance Cache, when a SmartFX effect uses both [PF\_OutFlag2\_AUTOMATIC\_WIDE\_TIME\_INPUT](#_bookmark178) &  [PF\_OutFlag\_NON\_PARAM\_VARY](#_bookmark147). Calling checkout\_layer during PF\_Cmd\_SMART\_PRE\_RENDER returns empty rects in PF\_CheckoutResult. The workaround is to simply make the call again. This workaround is no longer needed in 11.0.1. |

|  |  |
| --- | --- |
| Member | Purpose |
| result\_rect | The output (in layer coordinates) resulting from the render request (can be empty). This cannot be bigger than the input request rectangle (unless PF\_RenderOutputFlag\_RETURNS\_EXTRA\_PIXELS is set), but can be smaller. |
| max\_result\_rect | The maximum size the output could possibly be, if After Effects requested all of it. This must not vary depending on requested output size. |
| solid | Set this TRUE if every pixel in the output will be fully opaque. Set if possible; it enables certain optimizations. |
| reserved | Ignore. |
| flags | Currently, the only flag is PF\_RenderOutputFlag\_RETURNS\_EXTRA\_PIXELS, which tells After Effects that the smart effect will return more pixels than After Effects requested. |
| pre\_render\_data | Point this at any data that the effect would like to access during rendering. Effects can also allocate handles and store them in out\_data>frame\_data, as with regular (non-smart) effects. Since [*PF\_Cmd\_SMART\_PRE\_RENDER*](#_bookmark96) can be called with no corresponding [*PF\_Cmd\_SMART\_RENDER*](#_bookmark98), effects must never delete this data themselves; once the effect returns from [*PF\_Cmd\_SMART\_PRE\_RENDER*](#_bookmark96), After Effects owns this data and will dispose of it (using either the following function or a standard free call). |
| delete\_pre\_render\_data\_func | Point this to a function that will eventually be called to delete the  pre\_render\_data. |

**PRESERVE\_RGB\_OF\_ZERO\_ALPHA**

preserve\_rgb\_of\_zero\_alpha is used both as input to the effect, to tell it what to render, and as output from the effect, to describe the input it needs (as passed to the

checkout call). When preserve\_rgb\_of\_zero\_alpha is set in an input request, the effect must pass it recursively when making checkouts, otherwise prior effects and masking will

eliminate those pixels that the effect would reveal. Use of this is discouraged, though still supported in CS3 (8.0).

**RECTANGLES**

Effects must set both result rectangles accurately. After Effects' caching system relies upon them, incorrect values can cause many problems. If the plug-in returns a result\_rect

smaller than the request\_rect, that tells After Effects the pixels inside the request\_rect but outside the result\_rect are empty. Similarly, max\_result\_rect must encompass all non-zero pixels; the effect will never be asked to render anything outside this region. If there are pixels outside this rectangle, they will never be displayed.

Mis-sized output rectangles can cause problems as well. If these rectangles are too big, a loss of performance results. Not only will many empty pixels be cached (robbing the application of valuable memory), the effect may be unnecessarily asked to render large regions of

nothing. For this reason, the max\_result\_rect must be computed correctly, rather than set to some arbitrarily large size.

Both result\_rect and max\_result\_rect may vary depending on the effect's parameters, the current time, et cetera; they are valid only for the given invocation of the effect.

However, max\_result\_rect *cannot* depend on the specific render request. It must be the same no matter what portion of the output is requested by After Effects.

It is legal to return an empty result\_rect if the request\_rect doesn’t intersect the

effect's output pixels; no rendering need be done. After Effects may also call the effect with an empty request\_rect, meaning the effect is only being asked to compute the max\_result\_rect.

preserve\_rgb\_of\_zero\_alpha can influence the bounds computation process (both

result\_rect and max\_result\_rect) and must be respected if the effect behaves differently depending on this setting.

**THE “SIZE” OF A LAYER**

As with non-smart effects, each smart effect can arbitrarily shrink or expand its requested

input. They cannot depend on a fixed frame size, and the size of the input may change over time. For example, the user could apply an animated drop shadow to a layer, which would add pixels to different edges of the layer at different times, depending on the direction in

which the shadow is cast.

Some effects (for example, those which need to align one layer against another) need some notion of “size.” This could be defined two ways, each with advantages and disadvantages.

The size of the original layer, before any effects and downsampling are applied, is given

in\_data>width/height. As this value is unaffected by subsequent effects, it can act an

absolute reference for things like center points. However, this is not fool-proof, as the user could have applied a distortion or translation effect. Also, this value is available only for the layer to which the effect is applied, not other layer parameters.

...or...

Every layer input has a max\_result\_rect which encompasses all pixel data, in some sense the master “size” of a layer. It is available for all layers, but changes over time according to

previously applied effects, possibly in ways the user might not expect (as in the drop shadow example above).

Note that the ref\_width/height and max\_result\_rect for an input may be obtained without rendering, by calling checkout\_layer with an empty request\_rect. This is

fairly efficient, and can be useful if the layer “size” is needed first to determine exactly which pixels are required for rendering. This is an example of requesting a layer in pre-render and then never calling checkout\_layer (in this case, there are none).

**FLAG ON THE PLAY**

Normally, the max\_result\_rect of a given PF\_RenderRequest will be cropped to the bounds of any applied mask. However, if [PF\_OutFlag2\_REVEALS\_ZERO\_ALPHA](#_bookmark176) is set, the max\_result\_rect will be the size of the layer.

### PF\_CMD\_SMART\_RENDER

The effect will receive at most one *PF\_Cmd\_SMART\_RENDER* call for each pre-render.

Note that render may never be called at all. After Effects may have only wanted to to

perform some bounds computations, or it may have subsequently discovered that an effect's output is not needed at all (which can happen, for example, if the pre-render phase for a

track matte returns a rectangle that does not intersect the effect's output.) All effects must be able to handle Pre-Render without Render without leaking resources or otherwise

entering an unstable state. During *PF\_Cmd\_SMART\_RENDER*, the extra parameter points to a PF\_SmartRenderExtra.

**TABLE 44: PF\_SMARTRENDEREXTRA**

|  |  |
| --- | --- |
| Member | Purpose |
| PF\_SmartRenderInput | Consists of a [PF\_RenderRequest](#_bookmark407), the bitdepth, and a pointer to pre\_render\_data (allocated during [*PF\_Cmd\_SMART\_PRE\_RENDER*](#_bookmark405)). This PF\_SmartRenderInput is identical to that passed in the corresponding *PF\_Cmd\_SMART\_PRE\_RENDER*. |

**TABLE 44: PF\_SMARTRENDEREXTRA**

|  |  |
| --- | --- |
| Member | Purpose |
| PF\_SmartRenderCallbacks | PF\_Err checkout\_layer\_pixels( PF\_ProgPtr effect\_ref,  A\_long checkout\_idL, PF\_EffectWorld \*\*pixels);  This is used to actually access the pixels in layers checked out during *PF\_Cmd\_SMART\_PRE\_RENDER*. The returned PF\_EffectWorld is valid for duration of current command or until checked in.  You are only allowed to call checkout\_layer\_pixels only once with the checkout\_idL used earlier in *PF\_Cmd\_SMART\_PRERENDER*. There must be a one-to-one mapping between the number of checkouts made in *PF\_Cmd\_SMART\_PRERENDER* and *PF\_Cmd\_SMART\_RENDER*. To  call checkout\_layer\_pixels more than once on a layer, you should call [checkout\_layer](#_bookmark409) on the same layer again with a different unique checkout\_idL in *PF\_Cmd\_SMART\_PRERENDER* and then use that checkout\_idL to do another checkout\_layer\_pixels in *PF\_Cmd\_SMART\_RENDER*.  PF\_Err checkin\_layer\_pixels( PF\_ProgPtr effect\_ref,  A\_long checkout\_idL);  It isn’t necessary to call (After Effects cleans up all such checkouts when the effect returns from *PF\_Cmd\_SMART\_RENDER*), but useful to free up memory.  PF\_Err checkout\_output(  PF\_ProgPtr effect\_ref, PF\_EffectWorld \*\*output);  Retrieves the output buffer. Note that effects are not allowed to check out output until at least one input has been checked out (unless the effect has no inputs at all).  NOTE: For optimal memory usage, request the output as late as possible, and request inputs as few at a time as possible. |

**WHEN TO ACCESS LAYER PARAMETERS**

Parameters other than layer inputs may be freely checked out at any point. Layer inputs

must be accessed during [*PF\_Cmd\_SMART\_PRE\_RENDER*](#_bookmark96). However, you aren't required to actually *use* every input. If you check out a frame (or portion thereof) in [*PF\_Cmd\_SMART\_PRE\_RENDER*](#_bookmark96) and do not subsequently check it out in [*PF\_Cmd\_SMART\_RENDER*](#_bookmark98) , it need never be rendered, greatly improving performance.

**WAIT, GIMME THAT LAYER BACK!**

checkout\_layer\_pixels can only be called once with the checkout\_id used earlier in PreRender. There has to be a one-to-one mapping on the number of checkouts made in

PreRender and SmartRender. If you need to check out the pixels of a layer more than once, perhaps because of the structure of your code, just use more than one checkout\_id. In

PreRender, call checkout\_layer on the same layer with different unique checkout\_ids. Then in SmartRender, use a different one of those checkout\_ids each time checkout\_layer\_pixels is called in SmartRender.

5 : EFFECT UI & EVENTS

Effects can provide custom UI in two areas: (1) the Effect Controls Window (custom ECW UI), and (2) the Composition or Layer Windows (Custom Comp UI). Effects that use

custom UI should set [PF\_OutFlag\_CUSTOM\_UI](#_bookmark155) during [PF\_Cmd\_GLOBAL\_SETUP](#_bookmark74), and handle the PF\_Cmd\_EVENT selector.

Custom ECW UI allows an effect to provide a parameter with a customized control, which can be used either with standard parameter types or [arbitrary data parameters](#_bookmark335). Parameters that have a custom UI should set [PF\_PUI\_CONTROL](#_bookmark214) when [adding the parameter](#_bookmark283).

Custom Comp UI allows an effect to provide direct manipulation of the video in the Composition or Layer Windows. When the effect is selected, the Window can overlay

custom controls directly on the video, and can handle user interaction with those controls, to adjust parameters more quickly and naturally. Effects should register themselves to receive

events by calling PF\_REGISTER\_UI.

After Effects can send events to effects for user interface handling and parameter

management, integrating effects into its central message queue. While many events are sent in response to user input, After Effects also sends events to effects which manage arbitrary data parameters. The type of event is specified in [PF\_EventExtra](#_bookmark434)->e\_type and the

various events are described below.

**TABLE 45: EVENTS**

|  |  |
| --- | --- |
| **Event** | **Indicates** |
| *PF\_Event\_NEW\_CONTEXT* | The user created a new context (probably by opening a window) for events. The plug-in is allowed to store state information inside the context using the context handle. [PF\_EventUnion](#_bookmark454) contains valid context and type, but everything else should be ignored. |
| *PF\_Event\_ACTIVATE* | The user activated a new context (probably by bringing a window into the foreground). [PF\_EventUnion](#_bookmark454) is empty. |
| *PF\_Event\_DO\_CLICK* | The user clicked within the effect’s UI. [PF\_EventUnion](#_bookmark454) contains a PF\_DoClickEventInfo. Handle the mouse click and  respond, passing along drag info; see sample code), within a context. NOTE: As of 7.0, do *not* block until mouse-up; instead, rely on *PF\_Event\_DRAG*. |

|  |  |
| --- | --- |
| **Event** | **Indicates** |
| *PF\_Event\_DRAG* | Also a Click Event, [PF\_EventUnion](#_bookmark454) contains a PF\_DoClickEventInfo. Request this by returning send\_drag == TRUE from *PF\_Event\_DO\_CLICK*.  Do this so After Effects can see new data from the user’s changes. |
| *PF\_Event\_DRAW* | Draw! [PF\_EventUnion](#_bookmark454) contains a PF\_DrawEventInfo. |
| *PF\_Event\_DEACTIVATE* | The user has deactivated a context (probably by bringing another window into the foreground). PF\_EventUnion is empty. |
| *PF\_Event\_CLOSE\_CONTEXT* | A context has been closed by the user. PF\_EventUnion will be empty. |
| *PF\_Event\_IDLE* | A context is open but nothing is happening. PF\_EventUnion is empty. |
| *PF\_Event\_ADJUST\_CURSOR* | The mouse is over the plug-in’s UI. Set the cursor by changing the PF\_CursorType in the PF\_AdjustCursorEventInfo. Use OS-specific calls to implement a custom cursor; tell After Effects you’ve done so by setting PF\_CursorType to PF\_Cursor\_CUSTOM. Use an After Effects cursor whenever  possible to preserve interface continuity. |
| *PF\_Event\_KEYDOWN* | Keystroke. [PF\_EventUnion](#_bookmark454) contains a PF\_KeyDownEvent. |
| *PF\_Event\_MOUSE\_EXITED* | New in CS6. Notification that the mouse is no longer over a specific view (layer or comp only). |

**PF\_EVENTEXTRA**

This structure provide context information for the current event. After Effects passes a pointer to this structure in the extra parameter of the [entry point function](#_bookmark63). The PF\_EventUnion (sent in the PF\_EventExtra) varies with the event type, and contains information specific to that event.

**TABLE 46: PF\_EVENTEXTRA**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| contextH | Handle to the [PF\_Context](#_bookmark444). This drawing context is used with the [*Drawbot suites*](#_bookmark477) for drawing, and also for the [*UI callbacks*](#_bookmark498). |
| e\_type | Which [*event*](#_bookmark421) is occurring. |

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| u | A [PF\_EventUnion](#_bookmark454) containing information specific to the event. |
| effect\_win | A [PF\_EffectWindowInfo](#_bookmark446) about the event if it occurs within the effects window.  Otherwise, as of After Effects 5.0, effect\_win can be  replaced by a PF\_WindowUnion. This struct contains both a PF\_EffectWindowInfo and an PF\_ItemWindowInfo, which (for now) is simply the port rectangle for the item window. Replacement only occurs if PF\_USE\_NEW\_WINDOW\_UNION was defined during compilation; otherwise, it will continue to be just a PF\_EffectWindowInfo. |
| cbs | Pointer to [*UI callbacks*](#_bookmark498), which are needed to translate points between layer, composition, and screen coordinate systems. |
| evt\_in\_flags | Event Input Flags. This currently contains only one value,  PF\_EI\_DONT\_DRAW, which you should check before drawing! |
| evt\_out\_flags | One or more of the following, combined with a bitwise OR operation:  PF\_EO\_NONE  PF\_EO\_HANDLED\_EVENT tells After Effects you’ve handled the event.  PF\_EO\_ALWAYS\_UPDATE forces After Effects to rerender the composite in response to every click or drag; this is the same  behavior generated by ‘alt-scrubbing’ the parameter value.  PF\_EO\_NEVER\_UPDATE prevents After Effects from rerendering the composite until the user stops clicking and dragging.  PF\_EO\_UPDATE\_NOW tells After Effects to update the view immediately after the event returns after calling PF\_InvalidateRect |

PF\_Context details the event’s UI context.

**TABLE 47: PF\_CONTEXT**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| magic | Do not change. |
| w\_type | The window type. If you have Custom Comp and ECW UIs in the  same plug-in, this is the way to differentiate between them (what kind of masochist are you, anyway?).  PF\_Window\_COMP, PF\_Window\_LAYER, PF\_Window\_EFFECT |
| reserved\_flt | Do not change. |
| plugin\_state[4] | An array of 4 A\_longs which the plug-in can use to store state information for a given context. |
| reserved\_drawref | A DRAWBOT\_DrawRef for use with the [*Drawbot*](#_bookmark477) suites. |
| \*reserved\_paneP | Do not change. |

If an event occurs in the ECP, an PF\_EffectWindowInfo is sent in PF\_EventExtra.

**TABLE 48: PF\_EFFECTWINDOWINFO**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| index | This indicates which parameter in the effect window is being affected. The controls are numbered from 0 to the number of controls minus 1. |
| area | This indicates if the control title (PF\_EA\_PARAM\_TITLE) or the control itself (PF\_EA\_CONTROL) are being affected. The title is the area still visible when the parameter’s topic (“twirly”) is spun up. |
| current\_frame | A PF\_Rect indicating the full frame of the area occupied by the control. |
| param\_title\_frame | A PF\_Rect indicating the title area of the control. |
| horiz\_offset | A horizontal offset from the left side of the title area in which to draw into the title. |

**PF\_EVENTUNION**

The PF\_EventUnion in PF\_EventExtra is a union of the four following structures.

**CLICK**

A mouse click or drag occurred within the custom UI’s area.

**TABLE 49: PF\_DOCLICKEVENTINFO**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| when | The (OS-level) time at which the click occurred. |
| screen\_point | Where, in screen coordinates, the click occurred. For Custom Comp UI, these coordinates can be converted to composition coordinates  using the [*UI Callbacks*](#_bookmark498). See the CCU sample project for an example. |
| num\_clicks | The number of clicks that occurred. |
| modifiers | Which modifier keys (if any) were held down during click. |
| continue\_refcon[4] | An array of 4 A\_intptr\_t the plug-in can use to store information during a click-drag-drag sequence. |
| send\_drag | Set this flag to TRUE to indicate continued dragging. The next click event will then effectively be a drag event. |
| last\_time | Set when the drag event ends (the user has released the mouse button). |

**DRAW**

After Effects needs your custom UI to refresh. Note: when handling draw requests, use the image dimensions provided in [PF\_InData](#_bookmark111) (rather that the dimensions of your input layer, as you would during [*PF\_Cmd\_RENDER*](#_bookmark89)).

**TABLE 50: PF\_DRAWEVENTINFO**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| update\_rect | The rectangle in which to draw, in the context window’s coordinate system. These coordinates can be converted to different coordinate  systems using the [*UI Callbacks*](#_bookmark498). See the CCU sample project for an example. |
| depth | Pixel depth of the drawing context. |

**KEYDOWN**

The user pressed a key, and the effect’s UI is active. Use the macros in AE\_EffectUI.h to access and manipulate the key codes received.

**TABLE 51: PF\_KEYDOWNEVENT**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| when | Time at which the click occurred. |
| screen\_point | Screen coordinate of the mouse pointer when the key was pressed. For Custom Comp UI, these coordinates can be converted to composition coordinates using the [*UI Callbacks*](#_bookmark498). See the CCU sample project for an example. |
| key\_code | Either a character code (for printable characters, we use the unshifted upper case version; A not a, 7 not &), or a control code:  PF\_ControlCode\_Unknown PF\_ControlCode\_Space PF\_ControlCode\_Backspace PF\_ControlCode\_Tab PF\_ControlCode\_Return PF\_ControlCode\_Enter PF\_ControlCode\_Escape PF\_ControlCode\_F1  ...  PF\_ControlCode\_F24 PF\_ControlCode\_PrintScreen PF\_ControlCode\_ScrollLock PF\_ControlCode\_Pause PF\_ControlCode\_Insert PF\_ControlCode\_Delete PF\_ControlCode\_Home PF\_ControlCode\_End PF\_ControlCode\_PageUp PF\_ControlCode\_PageDown PF\_ControlCode\_Help PF\_ControlCode\_Clear PF\_ControlCode\_Left PF\_ControlCode\_Right PF\_ControlCode\_Up PF\_ControlCode\_Down PF\_ControlCode\_NumLock PF\_ControlCode\_Command PF\_ControlCode\_Option  PF\_ControlCode\_Alt = PF\_ControlCode\_Option PF\_ControlCode\_Control PF\_ControlCode\_Shift PF\_ControlCode\_CapsLock PF\_ControlCode\_ContextMenu |

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| modifiers | Which (if any) modifier keys were down during the key press.  PF\_Mod\_NONE  PF\_Mod\_CMD\_CTRL\_KEY (cmd on Mac, ctrl on Windows)  PF\_Mod\_SHIFT\_KEY PF\_Mod\_CAPS\_LOCK\_KEY  PF\_Mod\_OPT\_ALT\_KEY (option on Mac, alt on Windows)  PF\_Mod\_MAC\_CONTROL\_KEY |

**ADJUSTCURSOR**

The cursor has moved onto (but not off of) the effect’s custom UI, to allow the effect to change the cursor.

**TABLE 52: PF\_ADJUSTCURSOREVENTINFO**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| screen\_point | Screen coordinate of the mouse pointer. For Custom Comp UI, these coordinates can be converted to composition coordinates using the [*UI*](#_bookmark498) [*Callbacks*](#_bookmark498). See the CCU sample project for an example. |
| modifiers | What, if any, modifier keys were held down when the message was sent. |
| set\_cursor | Set this to your desired cursor, or PF\_Cursor\_CUSTOM if you have set the cursor yourself using OS-specific calls. See AE\_EffectUI.h for a complete enumeration of built-in cursors. If you don't want to override the cursor, set this to PF\_Cursor\_NONE, or simply ignore this  message. |

**ARBITRARY PARAMETERS EVENT**

After Effects needs your plug-in to manage it’s arbitrary data parameter(s). Though

arbitrary data types are not required for custom UI support, PF\_ArbParamsExtra follows the EventInfo model.

**TABLE 53: PF\_ARBPARAMSEXTRA**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| which\_function | A PF\_FunctionSelector indicating which function is called |
| id | Used by After Effects; will match the ID assigned to the arbitrary data type during *PF\_Cmd\_PARAM\_SETUP*. |

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| padding | Used for byte-alignment |
| u { |  |
| new\_func\_params dispose\_func\_params copy\_func\_params flat\_size\_func\_params flatten\_func\_params unflatten\_func\_params interp\_func\_params compare\_func\_params print\_size\_func\_params print\_func\_params scan\_func\_params  } | (One of these will be passed; see [*Arbitrary Data Parameters*](#_bookmark335)) |

**CUSTOM UI AND DRAWBOT**

Custom UI uses a composited drawing model using Drawbot. The Drawbot suites can be used for:

1. Basic 2D path drawing: Lines, Rect, Arc, Bezier
2. Stroking/Filling/Shading paths
3. Image drawing: Compositing an ARGB/BGRA buffer onto the surface
4. Pushing/popping surface state
5. Text drawing, if supplier supports it (clients should first check if text drawing is supported before actual drawing)

Drawing may only occur during PF\_Event\_DRAW (and not during PF\_Event\_DRAG or PF\_Event\_DO\_CLICK). To use Drawbot, first get the drawing reference by passing in

PF\_Context to a new suite call [PF\_GetDrawingReference](#_bookmark484). If a non-NULL drawing reference is returned, use it to get the supplier and surface references from [DRAWBOT\_DrawbotSuite](#_bookmark485).

The Drawbot suites include DRAWBOT\_DrawbotSuite, DRAWBOT\_SupplierSuite, DRAWBOT\_SurfaceSuite, DRAWBOT\_PathSuite.

**MAKE YOUR CUSTOM UI LOOK NOT SO “CUSTOM”**

Use the new [PF\_EffectCustomUIOverlayThemeSuite](#_bookmark495) to match the host application UI. Your users will thank you.

**REDRAWING**

In order to redraw a specific area of a pane, we recommend the following:

1. Call [PF\_InvalidateRect](#_bookmark352) from the effect. This will cause a lazy display redraw, and will update at the next available idle moment. This rect is in coordinates related to the

associated pane. Using a NULL rect will update the entire pane.

1. Set the [event outflag](#_bookmark440) to PF\_EO\_UPDATE\_NOW, which will cause an immediate draw event for the specified pane when the current event returns.

If an effect needs to update more than one window at a time, it should set [PF\_OutFlag\_REFRESH\_UI](#_bookmark157), which will cause a redraw of the entire ECW, comp, and layer windows.

**HIDPI AND RETINA DISPLAY SUPPORT**

To support HiDPI and Retina Displays, you can use offscreen images that are twice the size, and then use the [Transform](#_bookmark492) function to scale the image down in half before drawing it.

**PF\_EFFECTCUSTOMUISUITE**

Enables an effect to get the drawing reference. This is the first call needed to use Drawbot.

**TABLE 54: PF\_EFFECTCUSTOMUISUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetDrawingReference | Get the drawing reference.  PF\_GetDrawingReference(  const PF\_ContextH effect\_contextH, DRAWBOT\_DrawRef \*referenceP0); |

Using the Drawbot reference, get the supplier and surface references.

**TABLE 55: DRAWBOT\_DRAWBOTSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| GetSupplier | Get the supplier reference. Needed to use [DRAWBOT\_SupplierSuite](#_bookmark487).  GetSupplier(  DRAWBOT\_DrawRef in\_drawbot\_ref, DRAWBOT\_SupplierRef \*out\_supplierP); |
| GetSurface | Get the surface reference. Needed to use [DRAWBOT\_SurfaceSuite](#_bookmark490).  GetSurface(  DRAWBOT\_DrawRef in\_drawbot\_ref, DRAWBOT\_SurfaceRef \*out\_surfaceP); |

**DRAWBOT\_SUPPLIERSUITE**

Calls to create and release drawing tools, get default settings, and query drawing capabilities.

**TABLE 56: DRAWBOT\_SUPPLIERSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| NewPen | Create a new pen. Release this using [ReleaseObject](#_bookmark489).  NewPen(  DRAWBOT\_SupplierRef in\_supplier\_ref, const DRAWBOT\_ColorRGBA \*in\_colorP, float in\_size,  DRAWBOT\_PenRef \*out\_penP); |
| NewBrush | Create a new brush. Release this using [ReleaseObject](#_bookmark489).  NewBrush(  DRAWBOT\_SupplierRef in\_supplier\_ref, const DRAWBOT\_ColorRGBA \*in\_colorP, DRAWBOT\_BrushRef \*out\_brushP); |
| SupportsText | Check if current supplier supports text.  SupportsText(  DRAWBOT\_SupplierRef in\_supplier\_ref, DRAWBOT\_Boolean \*out\_supports\_textB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| GetDefaultFontSize | Get the default font size.  GetDefaultFontSize(  DRAWBOT\_SupplierRef in\_supplier\_ref, float \*out\_font\_sizeF); |
| NewDefaultFont | Create a new font with default settings. You can pass the default font size from GetDefaultFontSize. Release this using [ReleaseObject](#_bookmark489).  NewDefaultFont(  DRAWBOT\_SupplierRef in\_supplier\_ref, float in\_font\_sizeF,  DRAWBOT\_FontRef \*out\_fontP); |
| NewImageFromBuffer | Create a new image from buffer passed to in\_dataP. Release this using [ReleaseObject](#_bookmark489).  NewImageFromBuffer(  DRAWBOT\_SupplierRef in\_supplier\_ref, int in\_width,  int in\_height,  int in\_row\_bytes,  DRAWBOT\_PixelLayout in\_pl, const void \*in\_dataP,  DRAWBOT\_ImageRef \*out\_imageP);  DRAWBOT\_PixelLayout can be one of the following: kDRAWBOT\_PixelLayout\_24RGB, kDRAWBOT\_PixelLayout\_24BGR, kDRAWBOT\_PixelLayout\_32RGB, ARGB (A is ignored) kDRAWBOT\_PixelLayout\_32BGR, BGRA (A is ignored). kDRAWBOT\_PixelLayout\_32ARGB\_Straight, kDRAWBOT\_PixelLayout\_32ARGB\_Premul, kDRAWBOT\_PixelLayout\_32BGRA\_Straight, kDRAWBOT\_PixelLayout\_32BGRA\_Premul |
| NewPath | Create a new path. Release this using [ReleaseObject](#_bookmark489).  NewPath(  DRAWBOT\_SupplierRef in\_supplier\_ref, DRAWBOT\_PathRef \*out\_pathP); |
| SupportsPixelLayoutBGRA | A given Drawbot implementation can support multiple channel  orders, but will likely prefer one over the other. Use the following four callbacks to get the preferred channel order for any API that takes a DRAWBOT\_PixelLayout (e.g. NewImageFromBuffer).  SupportsPixelLayoutBGRA( DRAWBOT\_SupplierRef in\_supplier\_ref,  DRAWBOT\_Boolean \*out\_supports\_bgraPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PrefersPixelLayoutBGRA | PrefersPixelLayoutBGRA( DRAWBOT\_SupplierRef in\_supplier\_ref,  DRAWBOT\_Boolean \*out\_prefers\_bgraPB); |
| SupportsPixelLayoutARGB | SupportsPixelLayoutARGB( DRAWBOT\_SupplierRef in\_supplier\_ref,  DRAWBOT\_Boolean \*out\_supports\_argbPB); |
| PrefersPixelLayoutARGB | PrefersPixelLayoutARGB( DRAWBOT\_SupplierRef in\_supplier\_ref,  DRAWBOT\_Boolean \*out\_prefers\_argbPB); |
| RetainObject | Retain (increase reference count on) any object (pen, brush, path,  etc). For example, it should be used when any object is copied and the copied object should be retained.  RetainObject(  DRAWBOT\_ObjectRef in\_obj\_ref); |
| ReleaseObject | Release (decrease reference count on) any object (pen, brush, path, etc). This function MUST be called for any object created using NewXYZ() from this suite. Do not call this function on a  DRAWBOT\_SupplierRef and DRAWBOT\_SupplierRef, since these are not created by the plug-in.  ReleaseObject(  DRAWBOT\_ObjectRef in\_obj\_ref); |

**DRAWBOT\_SURFACESUITE**

Calls to draw on the surface, and to query and set drawing settings.

**TABLE 57: DRAWBOT\_SURFACESUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PushStateStack | Push the current surface state onto the stack. It should be popped to retrieve old state. It is required to restore state if you are going to clip or transform a surface or change the interpolation or anti-aliasing  policy.  PushStateStack(  DRAWBOT\_SurfaceRef in\_surface\_ref); |
| PopStateStack | Pop the last pushed surface state off the stack.  PopStateStack(  DRAWBOT\_SurfaceRef in\_surface\_ref); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PaintRect | Paint a rectangle with a color on the surface.  PaintRect(  DRAWBOT\_SurfaceRef in\_surface\_ref, const DRAWBOT\_ColorRGBA \*in\_colorP, const DRAWBOT\_RectF32 \*in\_rectPR); |
| FillPath | Fill a path using a brush and fill type.  FillPath(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_BrushRef in\_brush\_ref,  DRAWBOT\_PathRef in\_path\_ref,  DRAWBOT\_FillType in\_fill\_type);  DRAWBOT\_FillType is one of the following: kDRAWBOT\_FillType\_EvenOdd, kDRAWBOT\_FillType\_Winding |
| StrokePath | Stroke a path using a pen.  StrokePath(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_PenRef in\_pen\_ref,  DRAWBOT\_PathRef in\_path\_ref); |
| Clip | Clip the surface.  Clip(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_SupplierRef in\_supplier\_ref, const DRAWBOT\_Rect32 \*in\_rectPR); |
| GetClipBounds | Get clip bounds.  GetClipBounds(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_Rect32 \*out\_rectPR); |
| IsWithinClipBounds | Checks whether a rect is within the clip bounds.  IsWithinClipBounds(  DRAWBOT\_SurfaceRef in\_surface\_ref, const DRAWBOT\_Rect32 \*in\_rectPR, DRAWBOT\_Boolean \*out\_withinPB); |
| Transform | Transform the last surface state.  Transform(  DRAWBOT\_SurfaceRef in\_surface\_ref, const DRAWBOT\_MatrixF32 \*in\_matrixP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| DrawString | Draw a string.  DrawString(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_BrushRef in\_brush\_ref,  DRAWBOT\_FontRef in\_font\_ref, const DRAWBOT\_UTF16Char \*in\_stringP, const DRAWBOT\_PointF32 \*in\_originP, DRAWBOT\_TextAlignment in\_alignment\_style, DRAWBOT\_TextTruncation in\_truncation\_style, float in\_truncation\_width);  DRAWBOT\_TextAlignment is one of the following: kDRAWBOT\_TextAlignment\_Left, kDRAWBOT\_TextAlignment\_Center, kDRAWBOT\_TextAlignment\_Right  DRAWBOT\_TextTruncation is one of the following: kDRAWBOT\_TextTruncation\_None, kDRAWBOT\_TextTruncation\_End, kDRAWBOT\_TextTruncation\_EndEllipsis, kDRAWBOT\_TextTruncation\_PathEllipsis |
| DrawImage | Draw an image created using NewImageFromBuffer()on the surface. Alpha = [0.0f, 1.0f ].  DrawImage(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_ImageRef in\_image\_ref, const DRAWBOT\_PointF32 \*in\_originP, float in\_alpha); |
| SetInterpolationPolicy | SetInterpolationPolicy(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_InterpolationPolicy in\_interp);  DRAWBOT\_InterpolationPolicy is one of the following: kDRAWBOT\_InterpolationPolicy\_None, kDRAWBOT\_InterpolationPolicy\_Med, kDRAWBOT\_InterpolationPolicy\_High |
| GetInterpolationPolicy | GetInterpolationPolicy(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_InterpolationPolicy \*out\_interpP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| SetAntiAliasPolicy | SetAntiAliasPolicy(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_AntiAliasPolicy in\_policy);  DRAWBOT\_AntiAliasPolicy is one of the following: kDRAWBOT\_AntiAliasPolicy\_None, kDRAWBOT\_AntiAliasPolicy\_Med, kDRAWBOT\_AntiAliasPolicy\_High |
| GetAntiAliasPolicy | GetAntiAliasPolicy(  DRAWBOT\_SurfaceRef in\_surface\_ref, DRAWBOT\_AntiAliasPolicy \*out\_policyP); |
| Flush | Flush drawing. This is not always needed, and if overused, may cause excessive redrawing and flashing.  Flush(  DRAWBOT\_SurfaceRef in\_surface\_ref); |

**DRAWBOT\_PATHSUITE**

Calls to draw paths.

**TABLE 58: DRAWBOT\_PATHSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| MoveTo | Move to a point.  MoveTo(  DRAWBOT\_PathRef in\_path\_ref,  float in\_x,  float in\_y); |
| LineTo | Add a line to the path.  LineTo(  DRAWBOT\_PathRef in\_path\_ref,  float in\_x,  float in\_y); |
| BezierTo | Add a cubic bezier to the path.  BezierTo(  DRAWBOT\_PathRef in\_path\_ref, const DRAWBOT\_PointF32 \*in\_pt1P, const DRAWBOT\_PointF32 \*in\_pt2P, const DRAWBOT\_PointF32 \*in\_pt3P); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AddRect | Add a rect to the path.  AddRect(  DRAWBOT\_PathRef in\_path\_ref, const DRAWBOT\_RectF32 \*in\_rectPR); |
| AddArc | Add a arc to the path. Zero start degrees == 3 o'clock. Sweep is clockwise. Units for angle are in degrees.  AddArc(  DRAWBOT\_PathRef in\_path\_ref, const DRAWBOT\_PointF32 \*in\_centerP, float in\_radius,  float in\_start\_angle,  float in\_sweep); |
| Close | Close the path.  Close(  DRAWBOT\_PathRef in\_path\_ref); |

**PF\_EFFECTCUSTOMUIOVERLAYTHEMESUITE**

This suite should be used for stroking and filling paths and vertices on the Composition and Layer Windows. After Effects is using this suite internally, and we have made it available to make custom UI look consistent across effects. The foreground/shadow colors are computed based on the app brightness level so that custom UI is always visible regardless of the

application’s Brightness setting in the Preferences.

**TABLE 59: PF\_EFFECTCUSTOMUIOVERLAYTHEMESUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetPreferredForegroundColor | Get the preferred foreground color.  PF\_GetPreferredForegroundColor( DRAWBOT\_ColorRGBA \*foreground\_colorP); |
| PF\_GetPreferredShadowColor | Get the preferred shadow color.  PF\_GetPreferredShadowColor( DRAWBOT\_ColorRGBA \*shadow\_colorP); |
| PF\_GetPreferredStrokeWidth | Get the preferred foreground & shadow stroke width.  PF\_GetPreferredStrokeWidth(  float \*stroke\_widthPF); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PF\_GetPreferredVertexSize | Get the preferred vertex size.  PF\_GetPreferredVertexSize(  float \*vertex\_sizePF); |
| PF\_GetPreferredShadowOffset | Get the preferred shadow offset.  PF\_GetPreferredShadowOffset(  A\_LPoint \*shadow\_offsetP); |
| PF\_StrokePath | Stroke the path with the overlay theme foreground color.  Optionally draw the shadow using the overlay theme shadow color. Uses overlay theme stroke width for stroking foreground and shadow strokes.  PF\_StrokePath(  const DRAWBOT\_DrawRef drawbot\_ref, const DRAWBOT\_PathRef path\_ref PF\_Boolean draw\_shadowB); |
| PF\_FillPath | Fills the path with overlay theme foreground color. Optionally draw the shadow using the overlay theme shadow color.  PF\_FillPath(  const DRAWBOT\_DrawRef drawbot\_ref, const DRAWBOT\_PathRef path\_ref PF\_Boolean draw\_shadowB); |
| PF\_FillVertex | Fills a square vertex around the center point using the overlay theme foreground color and vertex size.  PF\_FillVertex(  const DRAWBOT\_DrawRef drawbot\_ref, const A\_FloatPoint \*center\_pointP PF\_Boolean draw\_shadowB); |

**UI CALLBACKS**

After Effects provides callbacks for transposing between coordinate systems, and obtaining OS-specific information about drawing contexts, without guesswork or asking the OS

directly. Use these callbacks! Pointers to these callbacks are provided in PF\_EventCallbacks. Use the macros in AE\_EffectUI.h and AE\_EffectCB.h to access these routines.

It is possible to build a functioning plug-in which utilizes a custom UI without

implementing the coordinate system transposition callbacks. However, the moment a user

zooms into the layer panel or rotates a layer, your plug-in will behave badly. We added these macros and callbacks so that custom user interfaces could be easily integrated into the After Effects UI, without inflicting user interface overhead on developers. Again, please use them!

These macros default the refcon and context handle for simplicity. The refcon assumes you have a local variable named “extra”. The default context is the current context. These

default parameters are defined in the PF\_EventCallbacks structure (in AE\_EffectUI.h).

You can override the defaults by accessing the callbacks through the PF\_EventExtra

structure. We don’t recommend (or support) modification of the macros in the header file. Don’t do it!

**TABLE 60: UI CALLBACKS**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| layer\_to\_comp | Transforms layer panel coordinates to the composition panel coordinates.  PF\_Err layer\_to\_comp ( void \*refcon, PF\_ContextH context,  A\_long curr\_time,  A\_long time\_scale, PF\_FixedPoint \*pt); |
| comp\_to\_layer | Transforms composition panel coordinates to the layer panel coordinates.  PF\_Err comp\_to\_layer ( void \*refcon, PF\_ContextH context,  A\_long curr\_time,  A\_long time\_scale, PF\_FixedPoint \*pt); |
| get\_comp2layer\_xform | Returns the matrix used to convert from the composition panel to the layer panel. If \*exists returns FALSE, the  matrix cannot be computed because the layer scales to zero.  PF\_Err get\_comp2layer\_xform ( void \*refcon, PF\_ContextH context, A\_long curr\_time,  long time\_scale,  long \*exists, PF\_FloatMatrix \*comp2layer); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| get\_layer2comp\_xform | Returns the transformation matrix used to convert from the layer panel to the composition panel. This always  exists.  PF\_Err get\_layer2comp\_xform ( void \*refcon, PF\_ContextH context, A\_long curr\_time,  A\_long time\_scale, PF\_FloatMatrix \*layer2comp); |
| source\_to\_frame | Transforms the source coordinates in the current context to screen coordinates. Screen (frame) coordinates are  affected by the current zoom level.  PF\_Err source\_to\_frame(  void \*refcon, PF\_ContextH context, PF\_FixedPoint \*pt); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| frame\_to\_source | Transforms the screen coordinates identified by \*pt to the source coordinates of the current context.  PF\_Err frame\_to\_source(  void \*refcon, PF\_ContextH context, PF\_FixedPoint \*pt); |
| PF\_GET\_PLATFORM\_DATA | Retrieves platform-specific data.For plug-ins loaded with localized resource files, PF\_PlatData\_RES\_FILE\_PATH will point to the external file, not the plug-in file. Use PF\_PlatData\_EXE\_FILE\_PATH if you want the  path of your plug-in.  Starting in CS6, use PF\_PlatData\_EXE\_FILE\_PATH\_W and PF\_PlatData\_RES\_FILE\_PATH\_W instead of the old non-wide calls.  PF\_Err PF\_GET\_PLATFORM\_DATA (  PF\_PlatDataID which, void \*ppData);  PF\_PlatDataID can have the following values:  PF\_PlatData\_MAIN\_WND PF\_PlatData\_EXE\_FILE\_PATH\_DEPRECATED PF\_PlatData\_RES\_FILE\_PATH\_DEPRECATED PF\_PlatData\_RES\_REFNUM //Mac OS PF\_PlatData\_RES\_DLLINSTANCE // Win PF\_PlatData\_BUNDLE\_REF PF\_PlatData\_EXE\_FILE\_PATH\_W // new CS6 PF\_PlatData\_RES\_FILE\_PATH\_W // new CS6 |

## TIPS AND TRICKS

**UI PERFORMANCE**

Experiment with [PF\_EO\_ALWAYS\_UPDATE](#_bookmark442) and [PF\_EO\_NEVER\_UPDATE](#_bookmark443), to find a happy medium between responsiveness and accuracy.

**NO MORE BLACK**

On Mac OS, the foreground and background colors are not set to white and black when

custom UI draw events are sent. This is by design; you don’t have to change the background color when you’re drawing directly into our context.

**HOW DEEP ARE MY PIXELS?**

There is no way to determine the bit depth of the layer(s) being processed during events.

However, you can cache the last-known pixel depth in your sequence data. Better still, you

can have your fixed and float slider parameters rely on the PF\_ValueDisplayFlags in their parameter definitions; if you use this, it will have your parameters’ UI respond to the user’s preferences for pixel display values. You can also check the depth of your input world during PF\_Cmd\_RENDER.

**ARBITRARY DATA**

An arbitrary data parameter is an excellent way to manage your custom UI. Store state,

preference, and last-item-used information in an arb, and you’ll always be able to recover it. After Effects manages parameters with a much richer message stream than custom UIs.

**CUSTOM UI IMPLEMENTATION FOR** **COLOR SAMPLING, USING KEYFRAMES**

A plug-in may want to get a color from a layer within a composition. The user would use the eyedropper associated with a color parameter, or the plug-in’s custom composition panel UI, to select the point. During the click event, the plug-in converts the coordinates of the click

into layer space, and stores that information in sequence data. It then forces a re-render, during which it has access to the color of the layer point corresponding to the stored

coordinates. The plug-in stores the color value in sequence data, and cancels the render,

requesting a redraw of the affected parameter(s). Finally, during the draw, the plug-in adds appropriate keyframes to its color parameter stream using the [KeyframeSuite](#_bookmark640). Yes, this

means the effect needs to [cheat](#_bookmark718) and use the AEGP API.

6 : AUDIO

After Effects can process audio encoded at up to 96Khz, floating point (24-bit) resolution, mono or stereo. We provide high quality resampling. PF\_InData and PF\_OutData both

contain information specific to audio handling.

While audio isn’t the focus of After Effects’ feature set, it is an important component of

compositing and pre-visualization workflows. Also, several engineers on our team are audio fanatics, and ensure that our audio effects (and the whole audio pipeline) are of the highest quality.

**GLOBAL OUTFLAGS**

All audio effects must set either PF\_OutFlag\_AUDIO\_EFFECT\_TOO or PF\_OutFlag\_AUDIO\_EFFECT\_ONLY. PF\_OutFlag\_I\_USE\_AUDIO is for visual effects that check out audio data, but don’t modify it. PF\_OutFlag\_AUDIO\_FLOAT\_ONLY,

PF\_OutFlag\_AUDIO\_IIR and PF\_OutFlag\_I\_SYNTHESIZE\_AUDIO provide greater control over audio output (see [PF\_OutFlags](#_bookmark143) for more details).

### AUDIO DATA STRUCTURES

The following data types are used by After Effects to describe audio data.

**TABLE 61: AUDIO DATA STRUCTURES**

|  |  |
| --- | --- |
| **Structure** | **Description** |
| PF\_SoundFormat | Indicates whether the audio is in unsigned pulse code modulation (PCM), signed PCM, or floating point format. |
| PF\_SoundSampleSize | Samples are in 1, 2, or 4 byte format. |
| PF\_SoundChannels | Indicates whether the audio is mono or stereo. |

**TABLE 61: AUDIO DATA STRUCTURES**

|  |  |
| --- | --- |
| **Structure** | **Description** |
| PF\_SoundFormatInfo | Contains the sampling rate, number of channels, sample size, and format of the audio to which it refers. |
| PF\_SoundWorld | Use PF\_SoundWorlds to represent audio. In addition to a PF\_SoundFormatInfo, they contain the length of the audio, and a pointer to the actual audio data. |

PF\_SoundFormat, PF\_SoundSampleSize, and PF\_SoundChannels are all contained

within a PF\_SoundFormatInfo. PF\_SoundWorlds contain a PF\_SoundFormatInfo, and further instance-specific information.

### AUDIO-SPECIFIC FLOAT SLIDER VARIABLES

PF\_Param\_FLOAT\_SLIDERs contain several parameters not found in other sliders; flags, phase, and curve tolerance.

**FLAGS**

The only flag available is PF\_FSliderFlag\_WANT\_PHASE. This registers the effect to receive updated phase information from After Effects during audio rendering. To

understand what this flag does, turn it off and check your output.

**PHASE**

This is where the requested phase value is stored.

**CURVE TOLERANCE**

Curve tolerance is used by After Effects to subdivide the audio effects’ time-variant parameters. Set this to zero for default behavior (or for non-audio FLOAT\_SLIDER parameters).

**WHAT’S ZERO, REALLY?**

When amplitude is zero, After Effects is at -192db.

### ACCESSING AUDIO DATA

Use [PF\_CHECKOUT\_LAYER\_AUDIO](#_bookmark294) to retrieve an audio layer. This layer is opaque; use [PF\_GET\_AUDIO\_DATA](#_bookmark296) to access specific details about that audio. As with pixel data, it’s important that you check in the audio as soon as possible.

If your effect requires as input a time span different from the output time span, update the

startsampL and endsampL field in PF\_OutData during [*PF\_Cmd\_AUDIO\_SETUP*](#_bookmark93).

**EXTENDING AUDIO CLIPS**

You cannot extend the length of an audio clip through the API. However, it is a relatively

simple matter for the user to extend the length of the clip before applying your effect. Apply time remapping to the layer and simply extend the out point. If you’re adding a delay effect to a sounds clip, you’d want to allow it time to fade away instead of truncating the sound at the original end point. Document the steps users should take when applying your effect.

### AUDIO CONSIDERATIONS

The After Effects audio API supports sampling rates up to 96Khz, in as many formats as possible. In the same way that plug-ins’ pixel manipulation functions should remain

“resolution independent”, audio plug-ins should be sample rate- and bit depth-independent.

Your plug-in can’t know anything about the final output format of the audio in question; it might get stretched, normalized, truncated, or phase-inverted between the application of

your plug-in and the final output.

Audio filters encounter different issues than do image filters. Investigate the SDK sample for one possible implementation of audio rendering.

# 7 : AEGPS

The After Effects General Plug-in (AEGP) API is powerful and broad, offering

functionality beyond what is available to effect plug-ins. To users, AEGPs appear to be part of After Effects. They can add, intercept, and trigger menu commands, access the keyframe database, and register functions as part of After Effects’ internal messaging. AEGPs can add and remove items to projects and compositions, add and remove filters and keyframes. Once its command is triggered, AEGPs use the numerous PICA function suites (described in this chapter) to work with every After Effects item.

AEGPs can publish function suites for plug-ins, manipulate all project elements, change

interpretations, replace files and determine which external files are used to render a project.

There are several specialized types of AEGP; Keyframers, Artisans, and I/O modules

(AEIOs). They are all still AEGPs, but have access to specialized messaging streams, for which they register with After Effects.

**WHAT’S NEW?**

For what’s new in CC releases, see [the heading in the Intro chapter](#_bookmark22).

**WHAT’S NEW IN CS6?**

3D is a major theme of AE CS6. A new AEGP\_LayerFlag\_ENVIRONMENT\_LAYER has been added. Many new [layer streams](#_bookmark624) were added. Additionally, AEGP\_LayerStream\_SPECULAR\_COEFF was renamed to AEGP\_LayerStream\_SPECULAR\_INTENSITY, AEGP\_LayerStream\_SHININESS\_COEFF was

renamed to AEGP\_LayerStream\_SPECULAR\_SHININESS, and

AEGP\_LayerStream\_METAL\_COEFF was renamed to just AEGP\_LayerStream\_METAL.

A new suite, [AEGP\_RenderQueueMonitorSuite](#_bookmark703), provides all the info a render queue manager needs to figure out what is happening at any point in a render.

[AEGP Mask Suite](#_bookmark653) is now at version 6, and provides functions to get and set the mask

feather falloff type. [AEGP Mask Outline Suite](#_bookmark659) is now at version 3, and provides access to get and set mask outline feather information.

[AEGP Comp Suite](#_bookmark575) is now at version 9. AEGP\_CreateTextLayerInComp and

AEGP\_CreateBoxTextLayerInComp now have a new parameter, select\_new\_layerB.

[AEGP Render Suite](#_bookmark684) is now at version 3, adding a new function to get the GUID for a render receipt.

Finally, we have added two new read-only [Dynamic Stream](#_bookmark637) flags: AEGP\_DynStreamFlag\_SHOWN\_WHEN\_EMPTY and AEGP\_DynStreamFlag\_SKIP\_REVEAL\_WHEN\_UNHIDDEN.

**OVERVIEW**

AEGPs use Plug-In Component Architecture (PICA) function suites to access all

functionality. They may also publish their own function suites, for use by effect plug-ins (since plug-in load order varies, AEGPs can’t depend on suites not provided by After

Effects). AEGPs can also request a suite and, if it’s not present, provide replacement functionality themselves.

**AEGP COMMUNICATION WITH AFTER EFFECTS**

For effect plug-ins, all communication with After Effects occurs through a single entry point function. This is not the case with AEGPs. While After Effects *does* call the entry point

function designated in the AEGP’s PiPL (which is still required), all subsequent

communication between After Effects and AEGPs is handled by the hook functions the AEGP registers. This registration must be performed from within the plug-in’s entry

function, using the [AEGP\_RegisterSuite](#_bookmark557).

**DIFFERENT TASKS, SAME API**

AEGPs work in the same manner, regardless of specialization. They can be simple, just

[adding one menu item](#_bookmark542) to trigger an external application, or complex like Artisans. While any plug-in can access any function suite, only plug-ins of the appropriate type will have access

to all the required parameters. Only Artisans will have render contexts, and only AEIO plug-ins will receive input and output specifications; messaging is dependent upon which hook functions are registered.

### DATA TYPES

Whenever possible, After Effects presents plug-ins with opaque data types, and provides

accessor functions for manipulating them. For example, video frames are represented using the opaque AEGP\_WorldH. While in some cases it might be more efficient to simply modify the underlying structure, by maintaining the opaqueness of the data types we allow for

changes to our implementation without making you recompile (and redistribute) your plug- ins.

**TABLE 62: AEGP API DATA TYPES**

|  |  |  |
| --- | --- | --- |
| **Type** | **Describes** | **Manage Using** |
| AEGP\_MemHandle | This structure contains more than just the referenced memory. So it should not be dereferenced directly. Use AEGP\_LockMemHandle in the AEGP Memory Suite to get a pointer to the memory referenced by the AEGP\_MemHandle. And of course, unlock it when you're done. | [*AEGP Memory Suite*](#_bookmark548) |
| AEGP\_ProjectH | The current After Effects project. Projects are a set of elements arranged hierarchically in a tree to preserve semantic relationships. Interior nodes of the tree are folders. As of CS6, there will only ever be one open project. | [*AEGP Proj Suite*](#_bookmark563) |
| AEGP\_ItemH | An abstraction describing any element of a project, including folders. An item is anything that can be  selected. Since multiple object types can be selected, we treat them as AEGP\_ItemHs until more specificity is  required. | [*AEGP Item Suite*](#_bookmark568) |
| AEGP\_Collection2H | A set of selected items. | [*AEGP Collection Suite*](#_bookmark570) |
| AEGP\_CompH | A composition is a sequence of renderable items that, together, produce output. A composition exists over a  time interval. Multiple compositions can exist within one project. | [*AEGP Comp Suite*](#_bookmark575) |
| AEGP\_FootageH | An item that can be rendered. Folders and compositions are the only items that are not footage. | [*AEGP Footage Suite*](#_bookmark581) |
| AEGP\_LayerH | An element of a composition. Layers are rendered in sequence, which allows for occlusions. Solids, text, paint, cameras, lights, images, and image sequences are all represented as layers.  Layers may be defined over sub-intervals of the composition's time interval. | [*AEGP Layer Suite*](#_bookmark592) |
| AEGP\_WorldH | A frame of pixels. | [*AEGP World Suite*](#_bookmark689) |

|  |  |  |
| --- | --- | --- |
| **Type** | **Describes** | **Manage Using** |
| AEGP\_EffectRefH | An effect applied to a layer. An effect is a function that takes as its argument a layer (and possibly other parameters) and returns an altered version of the layer for rendering. | [*AEGP Effect Suite*](#_bookmark608) |
| AEGP\_StreamRefH | Any [*parameter stream*](#_bookmark615) attached to a layer, in a composition. See the description of  [AEGP\_GetNewLayerStream](#_bookmark624) for a full list of stream types. | [*AEGP Stream*](#_bookmark622) [*Suite*](#_bookmark622), [*AEGP*](#_bookmark633)  [*Dynamic Stream*](#_bookmark633) [*Suite*](#_bookmark633), [*AEGP*](#_bookmark641)  [*Keyframe Suite*](#_bookmark641) |
| AEGP\_MaskRefH | A mask applied to a layer. An AEGP\_MaskRefH is used to access details about the mask stream, not the specific points which constitute the mask. A mask is a rasterized path (sequence of vertices) that partitions a layer into  two pieces, allowing each to be rendered differently. | [*AEGP Mask Suite*](#_bookmark653) |
| AEGP\_MaskOutline ValH | The specific points which constitute the mask.The points in a mask outline are ordered, and the mask need not be closed. | [*AEGP Mask Outline Suite*](#_bookmark659) |
| AEGP\_Text DocumentH | Represents the actual text associated with a text layer. | [*AEGP Text*](#_bookmark662)  [*Document Suite*](#_bookmark662) |
| AEGP\_Text OutlinesH | A reference to all the paths that make up the outlines of a given text layer. | [*AEGP Text Layer Suite*](#_bookmark664) |
| AEGP\_MarkerVal | The data associated with a given timeline marker. | [*AEGP Marker Suite*](#_bookmark650) |
| AEGP\_Persistent BlobH | A “ blob” of data containing the current preferences. | *AEGP Persistent Data Suite* |
| AEGP\_Render OptionsH | The settings associated with a render request. | [*AEGP Render Options Suite*](#_bookmark681) |
| AEGP\_LayerRender OptionsH | The settings associated with a layer render request. | [*AEGP Layer*](#_bookmark682)  [*Render Options Suite*](#_bookmark682) |
| AEGP\_Frame ReceiptH | A reference to a rendered frame. | [*AEGP Render Suite*](#_bookmark684) |
| AEGP\_RQItemRefH | An item in the render queue. | [*AEGP Render*](#_bookmark699) [*Queue Suite*](#_bookmark699),  *AEGP Render*  *Queue Item Suite* |
| AEGP\_OutputModul- eRefH | An output module, attached to a specific  AEGP\_RQItemRef in the render queue. | [*AEGP Output Module Suite*](#_bookmark706) |
| AEGP\_SoundDataH | The [*audio settings*](#_bookmark697) used for a given layer. | [*AEGP Sound Data Suite*](#_bookmark695) |

|  |  |  |
| --- | --- | --- |
| **Type** | **Describes** | **Manage Using** |
| AEGP\_RenderLayer ContextH | State information at the time of a render request, sent to an Artisan by After Effects. | [*AEGP Canvas Suite*](#_bookmark740) |
| AEGP\_Render ReceiptH | Used by Artisans when rendering. | [*AEGP Canvas Suite*](#_bookmark740) |

**NASTY, BRUTISH, AND SHORT**

Information about layers, streams, and many other items doesn’t survive long; it’s often

invalidated by user activity. Anything that modifies the quantity (not quality) of items will invalidate references to those items; adding a keyframe to a stream invalidates references to that stream, but forcing a layer to be rendered doesn’t invalidate references to it. Do not

cache layer pixels.

Caching references between calls to a specific hook function within your plug-in is not recommended; acquire information when you need it, and forget (release) it as soon as possible.

**WERE YOU JUST GOING TO LEAVE THAT DATA LYING AROUND?**

When you ask After Effects to populate and return handles to data structures, it’s important that you clean up after yourself. For the following data types, you must call the appropriate disposal routines.

**TABLE 63: DATA TYPES REQUIRING DISPOSAL**

|  |  |
| --- | --- |
| **Data Type** | **Disposal function** |
| AEGP\_Collection2H | [AEGP\_DisposeCollection](#_bookmark572) |
| AEGP\_FootageH | [AEGP\_DisposeFootage](#_bookmark584) |
| AEGP\_WorldH | [AEGP\_Dispose](#_bookmark691) (in [AEGP\_WorldSuite](#_bookmark689))  Or [AEGP\_DisposeTexture](#_bookmark744), if layer texture created using AEGP\_RenderTexture) |
| AEGP\_EffectRefH | [AEGP\_DisposeEffect](#_bookmark610) |
| AEGP\_MaskRefH | [AEGP\_DisposeMask](#_bookmark656) |
| AEGP\_RenderOptionsH | AEGP\_Dispose (in  [AEGP\_RenderOptionsSuite](#_bookmark681)) |

|  |  |
| --- | --- |
| **Data Type** | **Disposal function** |
| AEGP\_LayerRenderOptionsH | AEGP\_Dispose (in  [AEGP\_LayerRenderOptionsSuite](#_bookmark682)) |
| AEGP\_RenderReceiptH | AEGP\_DisposeRenderReceipt |

### IMPLEMENTATION

Because the functionality available through the AEGP API is so vast, and the integration with After Effects so complete, a good deal of design work is necessary to ensure that your plug-in behaves appropriately in all situations.

AEGPs interact with After Effects through PICA function suites. AEGPs are not loaded in a specific order. Check the version of the AEGP API (from within your AEGP’s entry point function) to confirm whether a given suite will be available. AEGPs may also use any effect API suite function which doesn’t require a PF\_ProgPtr (obtained by effects from PF\_InData).

**ENTRY POINT**

A\_Err AEGP\_PluginInitFuncPrototype( struct SPBasicSuite \*pica\_basicP, A\_long major\_versionL,

A\_long minor\_versionL,

AEGP\_PluginID aegp\_plugin\_id, AEGP\_GlobalRefcon \*global\_refconP)

The plug-in’s entry point, exported in the [PiPL resource](#_bookmark49), is called just once during launch;

all other calls to the AEGP go to the functions it’s registered. This is very different from the effect plug-in model, where all communication comes through the same entry point. Because plug-in load order may vary, it’s never a good idea to acquire suites not provided by After

Effects during your entry point function. Rather, wait until the appropriate hook function(s).

The AEGP API [version numbers](#_bookmark45) can help distinguish between different versions of After Effects, in case the AEGP needs to behave differently or handle different behavior.

**THE HOOK-UP**

Those other functions are registered as callback hooks. An AEGP that adds menu items must register an UpdateMenuHook function (with a function signature as described in AE\_GeneralPlug.h) which After Effects can call to determine whether or not to enable

those items. Similarly, plug-ins which process commands register a CommandHook (one for all commands).

**SPECIALIZATION**

AEIOs and Artisans must register with After Effects in order to receive the messaging

streams on which they depend. Like everything else in the AEGP API, this is done through a function suite; in this case, the aptly-named AEGP\_RegisterSuite.

**EXAMPLE: ADDING A MENU ITEM**

During your entry point function, use [CommandSuite>](#_bookmark551)[AEGP\_GetUniqueCommand()](#_bookmark553) to obtain a command ID from After Effects, for use with [AEGP\_InsertMenuCommand()](#_bookmark555). Use a different ID for each menu item you add.

Using AEGP\_RegisterSuite’s [AEGP\_RegisterCommandHook()](#_bookmark558), tell After Effects

which function to call when your menu item(s) are selected. The function you register using

[AEGP\_RegisterUpdateMenuHook()](#_bookmark559) enables and disabling your menu item(s). Your menu item(s) will be permanently disabled unless you register a menu updating function.

No matter how many menu items you add, you register only one CommandHook. When called, determine which menu item was chosen (based on the command ID), use AEGP

PICA suite functions to determine the current state of the project, and act accordingly. For example, keyframing plug-ins may want to disable their menu items unless a (keyframe-able) parameter stream is part of the current selection.

**PRIVATE DATA**

Unlike effects, AEGPs are never unloaded during an After Effects session. Still, that doesn’t mean that relying on static and global variables is a good idea.

All hook functions are passed a plugin\_refconPV for storage information specific to that

function. Many AEGP Suite functions take the aegp\_plugin\_id as a parameter; store it in the global\_refconPV you are passed, either in a structure you allocate or just the ID itself.

Where possible, use these refcons to store information, not statics and global variables. This becomes especially important when dealing with multi-threading issues.

Use global\_refconPV for your globals (like your aegp\_plugin\_id) and refcon for hook-function-specific storage.

A potential “multiple instances of After Effects” gotcha; when a second, command-line

instance of After Effects is launched, all of an AEGP’s handles are duplicated. If this causes problems (and it may), provide code that attaches saved handles to specific instantiations of your plug-in.

**THREADING**

AEGP supports no threading at all. Everything must be done from the main thread, either in response to a callback, or from the idle hook.

There is one call that is thread safe: [AEGP\_CauseIdleRoutinesToBeCalled](#_bookmark669)(). But

since SPBasicSuite itself is not thread safe, you'll need to stash off the function pointer in the main thread.

**AEGP SUITES**

As mentioned earlier, AEGPs do everything through suites. The following suites are used by all types of AEGPs, and may be called from within any hook function (except for the

RegisterSuite, which must be used from within the AEGP’s entry point). Following is a description of each function in every suite, and, where appropriate details on using those functions.

**TABLE 64: AEGP SUITES**

|  |  |
| --- | --- |
| **Suite** | **Description** |
| [Memory Suite](#_bookmark548) | Manage memory resources. Use this suite! Whenever memory-related errors are encountered, After Effects can report errors for you. |
| [Command Suite](#_bookmark551) | Manage your AEGP’s menu items. Used in conjunction with the  Register Suite. |
| [Register Suite](#_bookmark557) | Used in conjunction with the [Co](#_bookmark551)mmand Suite to add functions to menu commands. AEIOs and Artisans must use this suite’s functions to indicate to After Effects that they want to receive the appropriate message streams. You can replace some After Effects’ commands using this suite. |

|  |  |
| --- | --- |
| **Suite** | **Description** |
| [Project Suite](#_bookmark562) | Reads and modifies project data. |
| [Item Suite](#_bookmark566) | Manages items within a project or composition. Folders, Compositions, Solids, and Footage are all items. |
| [Collection Suite](#_bookmark570) | Query which items are currently selected, and create your own selection sets. It’s often a good UI move to select all the items your AEGP has modified, just to give the user some idea what you’ve done. |
| [Composition Suite](#_bookmark575) | Manages (and creates) compositions in a project, and composition-specific items like solids. |
| [Footage Suite](#_bookmark581) | Manages footage. |
| [Layer Suite](#_bookmark590) | Provides information about the layers within a composition, and the relationship(s) between the source and layer times. Solids, text, paint, cameras, lights, images, and image sequences can all become layers. |
| [Effect Suite](#_bookmark608) | Provides access to the effects applied to a layer. Use Stream suites to obtain effect keyframe information. Use  [AEGP\_EffectCallGeneric()](#_bookmark609) to communicate with effects that you setup ahead of time to respond to your AEGP. |
| [Stream Suite](#_bookmark622) | Used to access the values of a layer’s keyframe properties. |
| [Dynamic Stream Suite](#_bookmark633) | Used to access the characteristics of dynamic streams associated with a layer. |
| [Keyframe\_Suite](#_bookmark641) | Used to access and manipulate all keyframe data. |
| [Marker Suite](#_bookmark650) | Used to manipulate markers. Use [AEGP\_GetCompMarkerStream()](#_bookmark580) to get the composition marker stream. |
| [Mask Suite](#_bookmark651) | Provides access to retrieve information about a layer’s masks. |
| [Mask Outline Suite](#_bookmark659) | Used in conjunction with Stream Suite, this suite provides detailed information about the path rendered to make a layer’s mask. |
| [Text Document Suite](#_bookmark662) | Used to access the actual text on a text layer. |
| [Text Layer Suite](#_bookmark664) | Used to access the paths that make up the outlines of a text layer. |
| [Utility Suite](#_bookmark665) | Supplies error message handling, AEGP version checking and access to After Effects’ undo stack. |
| [Persistent Data Suite](#_bookmark672) | Query and manage all persistent data (i.e., the preferences file). AEGPs can also add their own data to the prefs. |
| [Color Settings Suite](#_bookmark675) | Obtain information on After Effects’ current color management settings. |
| Render Suite | Get rendered frames (and audio samples) from within an AEGP. |

|  |  |
| --- | --- |
| **Suite** | **Description** |
| [World Suite](#_bookmark689) | Allocate, dispose of, and query AEGP\_Worlds. Also provides a way to convert a PF\_EffectWorld into an AEGP\_World, for working with effect plug-ins. |
| Composite Suite | Exposes After Effects’ compositing functionality, including transfer modes, track matting, and good old fashioned bit copying. |
| [Sound Data Suite](#_bookmark695) | Functions for managing and accessing sound data. |
| [Render Queue Suite](#_bookmark698) | Add and remove items from the render queue. |
| [Render Queue Item Suite](#_bookmark701) | Query and modify items in the render queue. |
| [Render Options Suite](#_bookmark681) | Query and manage all items exposed in a render queue item’s options dialog. |
| [Output Module Suite](#_bookmark706) | Query and modify the output modules attached to items in the render queue. |
| [PF Interface Suite](#_bookmark710) | The functions in this suite, while technically part of the AEGP API, are for use by effects. |
| [AEGP Iterate Suite](#_bookmark714) | Gives AEGPs a way to have a function (which has the required signature) to be run on any or all available processors. |
| [File Import Manager Suite](#_bookmark716) | Registers AEGP file and project importers as part of After Effects’ file handling. |

**FAIL GRACEFULLY**

If a suite isn’t present, make every attempt to fail gracefully. Show the user a message

indicating the nature of the problem. Attempt to acquire and use an earlier version of the same suite.

Since AEGPs are so deeply integrated with After Effects, make sure that users know who or what is encountering a given problem. Identify yourself! Provide support and/or help

information to the user whenever possible.

**HANDLING HANDLES**

Use the AEGP Memory Suite to manage memory used by the AEGP. Whenever memory related errors are encountered, After Effects can report errors for you to find early on.

AEGP\_MemHandle is a structure that contains more than just the referenced memory. So it

should not be dereferenced directly. Use AEGP\_LockMemHandle to get a pointer to the memory referenced by the AEGP\_MemHandle. And of course, unlock it when you're done.

**TABLE 65: AEGP\_MEMORYSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewMemHandle | Create a new memory handle. This memory is guaranteed to be 16- byte aligned. plugin\_id is the ID passed in through the [*main*](#_bookmark538) [*entry point*](#_bookmark538), or alternatively what you obtained from [AEGP\_RegisterWithAEGP()](#_bookmark668). Use whatZ to identify the memory you are asking for. After Effects uses the string to display any related error messages.  AEGP\_NewMemHandle(  AEGP\_PluginID \*plugin\_id,  const A\_char \*whatZ,  AEGP\_MemSize size,  AEGP\_MemFlag flags,  AEGP\_MemHandle \*memPH); |
| AEGP\_FreeMemHandle | Release a handle you allocated using AEGP\_NewMemHandle().  AEGP\_FreeMemHandle( AEGP\_MemHandle memH); |
| AEGP\_LockMemHandle | Locks the handle into memory (cannot be moved by OS). Use this function prior to using memory allocated by AEGP\_NewMemHandle. Can be nested.  AEGP\_LockMemHandle( AEGP\_MemHandle memH,  void \*\*ptr\_to\_ptr); |
| AEGP\_UnlockMemHandle | Allows OS to move the referenced memory. Always balance lock calls with unlocks.  AEGP\_UnlockMemHandle(AEGP\_MemHandle memH); |
| AEGP\_GetMemHandleSize | Returns the allocated size of the handle.  AEGP\_GetMemHandleSize AEGP\_MemHandle memH,  AEGP\_MemSize \*sizeP); |
| AEGP\_ResizeMemHandle | Changes the allocated size of the handle.  AEGP\_ResizeMemHandle(  const char \*whatZ,  AEGP\_MemSize new\_size,  AEGP\_MemHandle memH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetMemReportingOn | If After Effects runs into problems with the memory handling, the error should be reported to the user. Make use of this during  development!  Only memory allocated and then leaked using this suite is reported using this call, so for example memory allocated using  [*PF\_HandleSuite1*](#_bookmark258) will not be reported.  AEGP\_SetMemReportingOn( A\_Boolean turn\_OnB); |
| AEGP\_GetMemStats | Obtain information about the number of currently allocated handles and their total size.  Only memory allocated using this suite is tracked and reported using this call, so for example memory allocated using  [*PF\_HandleSuite1*](#_bookmark258) will not be reported here.  AEGP\_GetMemStats(  AEGP\_MemID mem\_id,  A\_long \*countPL,  A\_long \*sizePL); |

**MANAGING MENU ITEMS**

Command Suites allow you to create and handle any menu events. To add your own menu commands, you must also use [AEGP\_RegisterSuite](#_bookmark557) to assign handlers to menu events.

**TABLE 66: AEGP\_COMMANDSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetUniqueCommand | Obtain a unique command identifier. Use the *Register Suite*  to register a handler for the command.  AEGP\_GetUniqueCommand(  AEGP\_Command \*unique\_commandP);  Note: On occasion After Effects will send command 0 (zero), so don’t use that as part of your command handling logic. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_InsertMenuCommand | Add a new menu command. Using nameZ = “-” will insert a separator. menu\_ID can be:  AEGP\_Menu\_NONE AEGP\_Menu\_APPLE AEGP\_Menu\_FILE AEGP\_Menu\_EDIT AEGP\_Menu\_COMPOSITION  AEGP\_Menu\_LAYER AEGP\_Menu\_EFFECT AEGP\_Menu\_WINDOW AEGP\_Menu\_FLOATERS  AEGP\_Menu\_KF\_ASSIST AEGP\_Menu\_IMPORT AEGP\_Menu\_SAVE\_FRAME\_AS AEGP\_Menu\_PREFS AEGP\_Menu\_EXPORT AEGP\_Menu\_ANIMATION  AEGP\_Menu\_PURGE  AEGP\_Menu\_NEW - Supported in CC and later  Locations can be set to a specific location in the menu or can be one assigned by After Effects:  AEGP\_MENU\_INSERT\_SORTED AEGP\_MENU\_INSERT\_AT\_BOTTOM AEGP\_MENU\_INSERT\_AT\_TOP  For AEGP\_Menu\_WINDOW, the BOTTOM and TOP options  haven’t been supported since CS4 and will return an error. We recommend SORTED.  AEGP\_InsertMenuCommand( AEGP\_Command command,  const A\_char \*nameZ,  AEGP\_MenuID menu\_id,  A\_long after\_itemL); |
| AEGP\_RemoveMenuCommand | Remove a menu command. If you were so motivated, you could remove ALL of the After Effects menu items.  AEGP\_RemoveMenuCommand( AEGP\_Command command); |
| AEGP\_SetCommandName | Set menu name of a command.  AEGP\_SetCommandName( AEGP\_Command command,  const A\_char \*nameZ); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_EnableCommand | Enable a menu command.  AEGP\_EnableCommand( AEGP\_Command command); |
| AEGP\_DisableCommand | Disable a menu command.  AEGP\_DisableCommand( AEGP\_Command command); |
| AEGP\_CheckMarkMenuCommand | After Effects will draw a check mark next to the menu command.  AEGP\_CheckMarkMenuCommand( AEGP\_Command command,  A\_Boolean checkB); |
| AEGP\_DoCommand | Call the handler for a specified menu command. Every After Effects menu item has an associated command; if your AEGP needs to call an After Effects menu item, you can use the  scripting command:  cmd = app.findMenuCommandId(text);  [or contact](mailto:zlam@adobe.com) *API Engineering* for the command number. Note that we make no guarantees that command IDs will be consistent from version to version.  AEGP\_DoCommand(AEGP\_Command command);  Here are a few command numbers that have been supplied to other developers, and may be of interest:  3061 Open selection, ignoring any modifier keys. 2285 RAM Preview.  2415 Play (spacebar).  2997 Crop composition to region of interest. 2372 Edit > Purge > Image Caches |

Register functions for After Effects’ use.

**TABLE 67: AEGP\_REGISTERSUITE5**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RegisterCommandHook | Register a hook (command handler) function with After Effects. If you are replacing a function which After Effects also handles, AEGP\_HookPriority determines whether your plug-in gets it first.  AEGP\_HP\_BeforeAE AEGP\_HP\_AfterAE  For each menu item you add, obtain your own AEGP\_Command using [AEGP\_GetUniqueCommand()](#_bookmark553) prior registering a single command\_hook\_func. Determine which command was sent within this hook function, and act accordingly.  Currently, AEGP\_HookPriority is ignored.  AEGP\_RegisterCommandHook(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_HookPriority hook\_priority, AEGP\_Command command, AEGP\_CommandHook command\_hook\_func void \*refconPV); |
| AEGP\_RegisterUpdateMenu Hook | Register your menu update function (which determines whether or not items are active), called every time any menu is to be drawn. This hook function handles updates for all menus.  AEGP\_RegisterUpdateMenuHook( AEGP\_PluginID aegp\_plugin\_id,  AEGP\_UpdateMenuHook update\_menu\_hook\_func, void \*refconPV); |
| AEGP\_RegisterDeathHook | Register your termination function. Called when the application quits.  AEGP\_RegisterDeathHook(  AEGP\_PluginID aegp\_plugin\_id,  AEGP\_DeathHook death\_hook\_func,  void \*refconPV); |
| AEGP\_RegisterVersionHook | Currently not called. |
| AEGP\_RegisterAboutString Hook | Currently not called. |
| AEGP\_RegisterAboutHook | Currently not called. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RegisterArtisan | [Register your Artisan. See the](#_bookmark731) Artisan chapter for more details.  AEGP\_RegisterArtisan(  A\_Version api\_version,  A\_Version Artisan\_version,  long aegp\_plugin\_id,  void \*aegp\_refconPV,  const A\_char \*match\_nameZ,  const A\_char \*Artisan\_nameZ, PR\_ArtisanEntryPoints \*entry\_funcsP); |
| AEGP\_RegisterIO | Register your AEIO plug-in. See the [*AEIO*](#_bookmark772) section for more details.  AEGP\_RegisterIO (  AEGP\_PluginID aegp\_plugin\_id,  AEGP\_IORefcon aegp\_refconP, const AEIO\_ModuleInfo \*io\_infoP,  const AEIO\_FunctionBlock4 \*aeio\_fcn\_blockP); |
| AEGP\_RegisterIdleHook | Register your IdleHook function. After Effects will call the function sporadically, while the user makes difficult artistic  decisions (or while they’re getting more coffee).  AEGP\_RegisterIdleHook(  AEGP\_PluginID aegp\_plugin\_id,  AEGP\_IdleHook idle\_hook\_func,  AEGP\_IdleRefcon refconP); |
| AEGP\_RegisterInteractive Artisan | Registers your AEGP as an interactive artisan, for use in  previewing and rendering all layers in a given composition.  AEGP\_RegisterInteractiveArtisan ( A\_Version api\_version,  A\_Version artisan\_version,  AEGP\_PluginID aegp\_plugin\_id,  void \*aegp\_refconPV,  const A\_char \*match\_nameZ,  const A\_char \*artisan\_nameZ, PR\_ArtisanEntryPoints \*entry\_funcsP); |
| AEGP\_RegisterPreset LocalizationString | Call this to register as many strings as you like for name-  replacement when presets are loaded. Any time a Property name is found, or referred to in an expression, and it starts with an ASCII tab character ('\t'), followed by one of the English names, it will be replaced with the localized name. (In English the tab character  will simply be removed).  AEGP\_RegisterPresetLocalizationString( const A\_char \*english\_nameZ, const A\_char \*localized\_nameZ); |

These functions access and modify project data. Support for multiple projects is included to prepare for future expansion; After Effects currently adheres to the single project model. To save project-specific data in After Effects’ preferences (and thus, outside the projects

themselves), use the [Persistent Data Suite](#_bookmark672). Use caution: the functions for opening

and creating projects do not save changes to the project currently open when they are called!

**TABLE 68: AEGP\_PROJSUITE6**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NumProjects | Currently will never return more than 1. After Effects can have only one project open at a time.  AEGP\_GetNumProjects)  A\_long \*num\_projPL) |
| AEGP\_GetIndProject | Retrieves a specific project by index.  AEGP\_GetProjectProjectByIndex( A\_long proj\_indexL,  AEGP\_ProjectH \*projPH); |
| AEGP\_GetProjectName | Get the project name (up to AEGP\_MAX\_PROJ\_NAME\_LEN +  1) in length.  AEGP\_GetProjectName( AEGP\_ProjectH projH,  A\_char \*nameZ); |
| AEGP\_GetProjectPath | Get the path of the project (empty string the project hasn’t been saved yet). The path is a handle to a NULL-terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetProjectPath( AEGP\_ProjectH projH,  AEGP\_MemHandle \*unicode\_pathPH) |
| AEGP\_GetProjectRootFolder | Get the root of the project, which After Effects also treats as a folder.  AEGP\_GetProjectRootFolder( AEGP\_ProjectH projH,  AEGP\_ItemH \*root\_folderPH) |
| AEGP\_SaveProjectToPath | Saves the entire project to the specified full path. The file path is a NULL-terminated UTF-16 string with platform separators.  AEGP\_SaveProjectToPath( AEGP\_ProjectH projH, const A\_UTF16Char \*pathZ); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetProjectTimeDisplay | Retrieves the current time display settings.  AEGP\_GetProjectTimeDisplay( AEGP\_ProjectH projH, AEGP\_TimeDisplay3 \*time\_displayP);  typedef struct {  AEGP\_TimeDisplayMode  display\_mode;  AEGP\_SourceTimecodeDisplayMode  footage\_display\_mode; A\_Boolean display\_dropframeB; A\_Boolean use\_feet\_framesB; A\_char timebaseC;  A\_char frames\_per\_footC;  AEGP\_FramesDisplayMode  frames\_display\_mode;  } AEGP\_TimeDisplay3;  enum {  AEGP\_TimeDisplay\_TIMECODE = 0, AEGP\_TimeDisplay\_FRAMES  };  typedef char AEGP\_TimeDisplayMode;  enum {  AEGP\_SourceTimecode\_ZERO= 0, AEGP\_SourceTimecode\_SOURCE\_TIMECODE  };  typedef char AEGP\_SourceTimecodeDisplayMode;  enum {  AEGP\_Frames\_ZERO\_BASED= 0, AEGP\_Frames\_ONE\_BASED, AEGP\_Frames\_TIMECODE\_CONVERSION  };  typedef char AEGP\_FramesDisplayMode; |
| AEGP\_SetProjectTimeDisplay | Specified the settings to be used for displaying time.  AEGP\_SetProjectTimeDisplay( AEGP\_ProjectH projH,  const AEGP\_TimeDisplay3 \*time\_displayP); |
| AEGP\_ProjectIsDirty | Returns TRUE if the project has been modified since it was opened.  AEGP\_ProjectIsDirty( AEGP\_ProjectH projH,  A\_Boolean \*is\_dirtyPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SaveProjectAs | Saves the project to the specified path. The file path is a NULL- terminated UTF-16 string with platform separators. NOTE: This will overwrite an existing file.  AEGP\_SaveProjectAs( AEGP\_ProjectH projH, const A\_UTF16Char \*pathZ); |
| AEGP\_NewProject | Creates a new project. NOTE: Will close the current project without saving it first!  AEGP\_NewProject( AEGP\_ProjectH\*new\_projectPH); |
| AEGP\_OpenProjectFromPath | Opens a project from the supplied path, and returns its AEGP\_ProjectH. The file path is a NULL-terminated UTF- 16 string with platform separators. NOTE: Will close the current project without saving it first!  AEGP\_OpenProjectFromPath(  const A\_UTF16Char \*pathZ, AEGP\_ProjectH \*projectPH); |
| AEGP\_GetProjectBitDepth | Retrieves the project bit depth.  AEGP\_GetProjectBitDepth( AEGP\_Projec tH projectH,  AEGP\_ProjBitDepth \*bit\_depthP);  AEGP\_ProjBitDepth will be one of the following:  AEGP\_ProjBitDepth\_8 AEGP\_ProjBitDepth\_16 AEGP\_ProjBitDepth\_32 |
| AEGP\_SetProjectBitDepth | Sets the project bit depth. Undoable.  AEGP\_SetProjectBitDepth( AEGP\_ProjectH projectH, AEGP\_ProjBitDepth bit\_depth); |

**TABLE 69: AEGP\_TIMEDISPLAY2**

|  |  |
| --- | --- |
| **Member** | **Description** |
| *Note: values in unused fields persist when After Effects is using a different display type.* | |
| AEGP\_TimeDisplayType type; | One of the following: AEGP\_TimeDisplayType\_TIMECODE AEGP\_TimeDisplayType\_FRAMES AEGP\_TimeDisplayType\_FEET\_AND\_FRAMES |

|  |  |
| --- | --- |
| **Member** | **Description** |
| A\_char timebaseC; | 0 - 100. Only used for  AEGP\_TimeDisplayType\_TIMECODE. |
| A\_Boolean non\_drop\_30B; | When the timebase is 30 and the item’s framerate is  29.97, determines whether to display as non-drop frame. |
| A\_char frames\_per\_footC; | Only used for  AEGP\_TimeDisplayType\_FEET\_AND\_FRAMES. |
| A\_long starting\_frameL; | Usually 0 or 1. Not used when type is usually 0 or 1, not used for AEGP\_TimeDisplayType\_TIMECODE. |
| A\_Boolean auto\_timecode\_baseB; | If TRUE, the project timecode display setting is set to auto. |

**CONTROL ITEMS WITHIN PROJECTS**

Accesses and modifies items within a project or composition. Anything in the project bin is an AEGP\_Item. Note that cameras have no source, and thus have no AEGP\_ItemH. Unless more specificity is required for the function(s) you’re using, remain as abstract as possible; AEGP\_Comps are passed into and returned from most functions as AEGP\_Items.

**TABLE 70: AEGP\_ITEMSUITE8**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetFirstProjItem | Retrieves the first item in a given project.  AEGP\_GetFirstProjItem( AEGP\_ProjectH projectH,  AEGP\_ItemH \*itemPH); |
| AEGP\_GetNextProjItem | Retrieves the next project item; \*next\_itemPH will be  NULL after the last item.  AEGP\_GetNextProjItem( AEGP\_ProjectH projectH, AEGP\_ItemH itemH,  AEGP\_ItemH \*next\_itemPH); |
| AEGP\_GetActiveItem | If the Project window is active, the active item is the selected item (if only one item is selected). If a  Composition, Timeline, or Footage window is active,  returns the parent of the layer associated with the front-  most tab in the window. Returns NULL if no item is active.  AEGP\_GetActiveItem( AEGP\_ItemH \*itemPH, |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_IsItemSelected | Returns true if the Project window is active and the item is selected.  AEGP\_IsItemSelected( AEGP\_ItemH itemH,  A\_Boolean \*selectedPB) |
| AEGP\_SelectItem | Toggles the selection state of the item, and (depending on deselect\_othersB) can deselect other items. This call selects items in the Project panel. To make selections in the Composition panel, use [AEGP\_SetSelection](#_bookmark579) in the  AEGP Comp Suite.  AEGP\_SelectItem( AEGP\_ItemH itemH,  A\_Boolean selectB,  A\_Boolean deselect\_othersB); |
| AEGP\_GetItemType | Gets type of an item. Note: solids don’t appear in the project, but can be the source to a layer.  AEGP\_GetItemType( AEGP\_ItemH itemH,  AEGP\_ItemType \*item\_typeP);  Items are one of the following types:  AEGP\_ItemType\_NONE AEGP\_ItemType\_FOLDER AEGP\_ItemType\_COMP AEGP\_ItemType\_SOLID AEGP\_ItemType\_FOOTAGE |
| AEGP\_GetTypeName | Get name of type. (name length up to  AEGP\_MAX\_TYPE\_NAME\_LEN + 1).  AEGP\_GetTypeName(  AEGP\_ItemType item\_type, A\_char \*nameZ); |
| AEGP\_GetItemName | Get item name. (name length has no limit). unicode\_namePH points to A\_UTF16Char (contains null terminated UTF16 string). It must be disposed with AEGP\_FreeMemHandle .  AEGP\_GetItemName(  AEGP\_PluginID pluginID,  AEGP\_ItemH itemH, AEGP\_MemHandle \*unicode\_namePH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetItemName | Specifies the name of the AEGP\_ItemH. (name length has no limit). Undoable.  AEGP\_SetItemName(  AEGP\_ItemH itemH, const A\_UTF16Char \*nameZ); |
| AEGP\_GetItemID | Returns the item’s unique ID, which persists across saves and loads of the project.  AEGP\_GetItemID(  AEGP\_ItemH itemH,  A\_long \*item\_idPL); |
| AEGP\_GetItemFlags | Get properties of an item.  AEGP\_GetItemFlags( AEGP\_ItemH itemH,  AEGP\_ItemFlags \*item\_flagsP);  Flag values (may be OR’d together):  AEGP\_ItemFlag\_MISSING AEGP\_ItemFlag\_HAS\_PROXY AEGP\_ItemFlag\_USING\_PROXY AEGP\_ItemFlag\_MISSING\_PROXY AEGP\_ItemFlag\_HAS\_VIDEO AEGP\_ItemFlag\_HAS\_AUDIO AEGP\_ItemFlag\_STILL  AEGP\_ItemFlag\_HAS\_ACTIVE\_AUDIO  Unlike the HAS\_AUDIO flag, this bit flag will set only if the comp has at least one layer where audio is actually on. |
| AEGP\_SetItemUseProxy | Toggle item’s proxy usage. Undoable.  AEGP\_SetItemUseProxy( AEGP\_ItemH itemH,  A\_Boolean use\_proxyB); |
| AEGP\_GetItemParentFolder | Get folder containing item.  AEGP\_GetItemParentFolder( AEGP\_ItemH itemH,  AEGP\_ItemH \*parent\_itemPH); |
| AEGP\_SetItemParentFolder | Sets an item’s parent folder. Undoable.  AEGP\_SetItemParentFolder( AEGP\_ItemH itemH,  AEGP\_ItemH parent\_folderH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetItemDuration | Get duration of item, in seconds.  AEGP\_GetItemDuration( AEGP\_ItemH itemH,  A\_Time \*durationPT); |
| AEGP\_GetItemCurrentTime | Get current time within item. Not updated while rendering.  AEGP\_GetItemCurrentTime( AEGP\_ItemH itemH,  A\_long \*curr\_timePT); |
| AEGP\_GetItemDimensions | Get width and height of item.  AEGP\_GetItemDimensions( AEGP\_ItemH itemH,  A\_long \*widthPL)  A\_long \*heightPL); |
| AEGP\_GetItemPixelAspectRatio | Get the width of a pixel, assuming its height is 1.0, as numerator over denominator.  AEGP\_GetItemPixelAspectRatio( AEGP\_ItemH itemH,  A\_Ratio \*ratioPRt); |
| AEGP\_DeleteItem | Removes item from all compositions. Undo-able. Do not use the AEGP\_ItemH after calling this function.  AEGP\_DeleteItem(  AEGP\_ItemH itemH); |
| AEGP\_GetItemSolidColor | Removed in AEGP\_ItemSuite4. See  [AEGP\_GetSolidFootageColor](#_bookmark586)  Given a solid item, return its color.  AEGP\_GetItemSolidColor( AEGP\_ItemH itemH,  PF\_Pixel \*PF\_Pixel); |
| AEGP\_SetSolidColor | Removed in AEGP\_ItemSuite4. See  [AEGP\_SetSolidFootageColor](#_bookmark587).  Sets the color of an existing solid (error if itemH is not a solid).  AEGP\_SetSolidColor( AEGP\_ItemH itemH, AEGP\_ColorVal color); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetSolidDimensions | Removed in AEGP\_ItemSuite4. See  [AEGP\_SetSolidFootageDimensions](#_bookmark588).  Sets the dimensions of an existing solid (error if itemH is not a solid).  AEGP\_SetSolidDimensions( AEGP\_ItemH itemH,  A\_short widthS,  A\_short heightS); |
| AEGP\_CreateNewFolder | Creates a new folder in the project. The newly created folder is allocated and owned by After Effects. Passing NULL for parent\_folderH0 creates the folder at the project’s root.  AEGP\_CreateNewFolder(  const A\_UTF16Char \*nameZ, AEGP\_ProjectH projH),  AEGP\_ItemH parentH0),  AEGP\_ItemH \*new\_folderPH); |
| AEGP\_SetItemCurrentTime | Sets the current time within a given itemH.  AEGP\_SetItemCurrentTime( AEGP\_ItemH itemH, const A\_Time \*new\_timePT); |
| [AEGP\_RenderNewItemSoundData](#_bookmark687)*() used to be here, but is now part of AEGP\_RenderSuite.* | |
| AEGP\_GetItemCommentLength | Retrieves the length (in characters) of the itemH’s comment.  AEGP\_GetItemCommentLength( AEGP\_ItemH itemH,  A\_u\_long \*buf\_sizePLu); |
| AEGP\_GetItemComment | Retrieves the itemH’s comment.  AEGP\_GetItemComment( AEGP\_ItemH itemH,  A\_u\_long buf\_sizeLu,  A\_char \*commentZ); |
| AEGP\_SetItemComment | Sets the itemH’s comment.  AEGP\_SetItemComment( AEGP\_ItemH itemH,  const char \*commentZ); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetItemLabel | Retrieves an item’s label.  AEGP\_GetItemLabel( AEGP\_ItemH itemH,  AEGP\_LabelID \*labelP); |
| AEGP\_SetItemLabel | Sets an item’s label.  AEGP\_SetItemLabel( AEGP\_ItemH itemH, AEGP\_LabelID label); |
| AEGP\_GetItemMRUView | Gets an item’s most recently used view. The view can be used with two calls in the AEGP\_ColorSettingsSuite, to  perform a color transform on a pixel buffer from working to view color space.  AEGP\_SetItemLabel( AEGP\_ItemH itemH,  AEGP\_ItemViewP \*mru\_viewP); |

**MANAGING SELECTIONS**

This suite manages selection states, mirroring the functionality supplied by vectors in the C++ Standard Template Library. Many types of items may be simultaneously selected in After Effects; AEGP\_CollectionItems are unions of layer, mask, effect, stream, mask

vertex, and keyframe items. First acquire the current collection, then iterate across its members to ensure that whatever your AEGP does is applicable to each.

We’ve added AEGP\_Collection2H and AEGP\_CollectionItemV2 so that selected dynamic streams can be handled with the AEGP\_CollectionSuite.

**TABLE 71: AEGP\_COLLECTIONSUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewCollection | Creates and returns a new, empty collection. To obtain the current composition’s selection as a collection, use AEGP\_GetNewCollection FromCompSelection.  AEGP\_NewCollection(  AEGP\_PluginID plugin\_id, AEGP\_Collection2H \*collectionPH); |
| AEGP\_DisposeCollection | Disposes of a collection.  AEGP\_DisposeCollection( AEGP\_Collection2H collectionH); |
| AEGP\_GetCollectionNumItems | Returns the number of items contained in the given collection.  AEGP\_GetCollectionNumItems( AEGP\_Collection2H collectionH, A\_u\_long \*num\_itemsPL); |
| AEGP\_GetCollectionItemByIndex | Retrieves (creates and populates) the index’d collection item.  AEGP\_GetCollectionItemByIndex( AEGP\_Collection2H collectionH, A\_u\_long indexL, AEGP\_CollectionItemV2 \*itemP); |
| AEGP\_CollectionPushBack | Adds an item to the given collection.  AEGP\_CollectionPushBack( AEGP\_Collection2H collectionH, const AEGP\_CollectionItemV2  \*itemP); |
| AEGP\_CollectionErase | Removes an index’d item (or items) from a given collection. NOTE: this range is exclusive, like STL iterators. To erase the first item, you would pass 0 and 1, respectively.  AEGP\_CollectionErase( AEGP\_Collection2H collectionH, A\_u\_long index\_firstL,  A\_u\_long index\_lastL); |

When AEGP\_StreamRefHs are inserted into a collection, they are adopted by the collection; do not free them. AEGP\_EffectRefHs, on the other hand, are not adopted, and must be

freed by the calling AEGP.

**MANIPULATE COMPOSITIONS**

Provide information about the compositions in a project, and create cameras, lights, and solids.

**TABLE 72:** **AEGP\_COMPSUITE10**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompFromItem | Retrieves the handle to the composition, given an item  handle. Returns NULL if itemH is not an AEGP\_CompH.  AEGP\_GetCompFromItem( AEGP\_ItemH itemH,  AEGP\_CompH \*compPH); |
| AEGP\_GetItemFromComp | Used to get the item handle, given a composition handle.  AEGP\_GetItemFromComp( AEGP\_CompH compH,  AEGP\_ItemH \*itemPH); |
| AEGP\_GetCompDownsampleFactor | Returns current downsample factor. Measured in pixels X by  Y. Users can choose a custom downsample factor with independent X and Y.  AEGP\_GetCompDownsampleFactor( AEGP\_CompH compH,  AEGP\_DownsampleFactor \*dsfP); |
| AEGP\_SetCompDownsampleFactor | Sets the composition’s downsample factor.  AEGP\_SetCompDownsampleFactor( AEGP\_CompH compH, AEGP\_DownsampleFactor \*dsfP); |
| AEGP\_GetCompBGColor | Returns the composition background color.  AEGP\_GetCompBGColor( AEGP\_CompH compH,  AEGP\_ColorVal \*bg\_colorP); |
| AEGP\_SetCompBGColor | Sets a composition’s background color.  AEGP\_SetCompBGColor(  AEGP\_CompH compH,  const AEGP\_ColorVal \*bg\_colorP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompFlags | Returns composition flags, or’d together.  AEGP\_GetCompFlags( AEGP\_CompH compH,  AEGP\_CompFlags \*AEGP\_CompFlags);  AEGP\_CompFlag\_SHOW\_ALL\_SHY AEGP\_CompFlag\_ENABLE\_MOTION\_BLUR AEGP\_CompFlag\_ENABLE\_TIME\_FILTER AEGP\_CompFlag\_GRID\_TO\_FRAME AEGP\_CompFlag\_GRID\_TO\_FIELDS AEGP\_CompFlag\_USE\_LOCAL\_DSF AEGP\_CompFlag\_DRAFT\_3D AEGP\_CompFlag\_SHOW\_GRAPH |
| AEGP\_GetShowLayerNameOr  -SourceName | New in CC. Passes back true if the Comp’s timeline shows layer names, false if source names. This will open the comp as a side effect.  AEGP\_GetShowLayerNameOrSourceName( AEGP\_CompH compH,  A\_Boolean \*layer\_names\_shownPB); |
| AEGP\_SetShowLayerNameOr  -SourceName | New in CC. Pass in true to have the Comp’s timeline show  layer names, false for source names. This will open the comp as a side effect.  AEGP\_SetShowLayerNameOrSourceName( AEGP\_CompH compH,  A\_Boolean \*layer\_names\_shownPB); |
| AEGP\_GetShowBlendModes | New in CC. Passes back true if the Comp’s timeline shows blend modes column, false if hidden. This will open the  comp as a side effect.  AEGP\_GetShowBlendModes( AEGP\_CompH compH,  A\_Boolean \*blend\_modes\_shownPB); |
| AEGP\_SetShowBlendModes | New in CC. Pass in true to have the Comp’s timeline show the blend modes column, false to hide it. This will open the comp as a side effect.  AEGP\_GetCompFlags( AEGP\_CompH compH,  A\_Boolean show\_blend\_modesB); |
| AEGP\_GetCompFramerate | Returns the composition’s frames per second.  AEGP\_GetCompFramerate( AEGP\_CompH compH,  A\_FpLong \*fpsPF); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetCompFramerate | Sets the composition’s frames per second.  AEGP\_SetCompFramerate( AEGP\_CompH compH,  A\_FpLong \*fpsPF); |
| AEGP\_GetCompShutterAnglePhase | The composition shutter angle and phase.  AEGP\_GetCompShutterAnglePhase( AEGP\_CompH compH,  A\_Ratio \*angle,  A\_Ratio \*phase); |
| AEGP\_GetCompShutterFrameRange | The duration of the shutter frame, in seconds.  AEGP\_GetCompShutterFrameRange( AEGP\_CompH compH,  const A\_Time \*comp\_timeP); |
| AEGP\_GetCompSuggestedMotion BlurSamples | Retrieves the number of motion blur samples After Effects will perform in the given composition.  AEGP\_GetCompSuggestedMotionBlurSamples( AEGP\_CompH compH,  A\_long \*samplesPL) |
| AEGP\_SetCompSuggestedMotion BlurSamples | Specifies the number of motion blur samples After Effects will perform in the given composition. Undoable.  AEGP\_SetCompSuggestedMotionBlurSamples( AEGP\_CompH compH,  A\_long samplesL); |
| AEGP\_GetCompMotionBlur- AdaptiveSampleLimit | New in CC. Retrieves the motion blur adaptive sample limit for the given composition. As of CC, a new comp defaults to 128.  AEGP\_GetCompMotionBlurAdaptiveSampleLimit( AEGP\_CompH compH,  A\_long \*samplesPL) |
| AEGP\_SetCompMotionBlur- AdaptiveSampleLimit | New in CC. Specifies the motion blur adaptive sample limit for the given composition. As of CC, both the limit and the suggested values are clamped to [2,256] range and the limit value will not be allowed less than the suggested value.  Undoable.  AEGP\_SetCompMotionBlurAdaptiveSampleLimit( AEGP\_CompH compH,  A\_long samplesL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompWorkAreaStart | Get the time where the current work area starts.  AEGP\_GetCompWorkAreaStart( AEGP\_CompH compH,  A\_Time \*startPT); |
| AEGP\_GetCompWorkAreaDuration | Get the duration of a composition’s current work area, in seconds.  AEGP\_GetCompWorkAreaDuration( AEGP\_CompH compH,  A\_Time \*durationPT); |
| AEGP\_SetCompWorkArea StartAndDuration | Set the work area start and duration, in seconds. Undo-able.  One call to this function is sufficient to set the layer’s in  point and duration; it’s not necessary to call it twice, once for each timespace.  AEGP\_SetCompWorkAreaStartAndDuration) AEGP\_CompH compH,  const A\_Time \*startPT)  const A\_Time \*durationPT); |
| AEGP\_CreateSolidInComp | Creates a new solid with a specified width, height, color, and duration in the composition. Undo-able.  If you pass NULL for the duration, After Effects uses its  preference for the duration of a new still. If you pass NULL, or an invalid time scale, duration is set to the length of the composition.  AEGP\_CreateSolidInComp(  const A\_UTF16Char \*utf\_nameZ, A\_Long widthL,  A\_Long heightL, const PF\_Pixel \*color, AEGP\_CompH parent\_compH,  const A\_Time \*durationPT0,  AEGP\_LayerH \*new\_solidPH); |
| AEGP\_CreateCameraInComp | Creates and adds a camera to the specified composition. Once created, you can manipulate the camera’s parameter streams using the [AEGP\_StreamSuite](#_bookmark622).  To specify a two-node camera, use [AEGP\_SetLayerFlag](#_bookmark596)  to set AEGP\_LayerFlag\_LOOK\_AT\_POI.  AEGP\_CreateCameraInComp(  const A\_UTF16Char \*utf\_nameZ, A\_FloatPoint center\_point,  AEGP\_CompH parent\_compH,  AEGP\_LayerH \*new\_cameraPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_CreateLightInComp | Creates and adds a light to the specified composition. Once created, you can manipulate the light’s parameter streams  using the [AEGP\_StreamSuite](#_bookmark622).  AEGP\_CreateLightInComp(  const A\_UTF16Char \*utf\_nameZ, A\_FloatPoint center\_point,  AEGP\_CompH parent\_compH,  AEGP\_LayerH \*new\_lightPH); |
| AEGP\_CreateComp | Creates a new composition for the project. If you don’t provide a parent folder, the composition will be at the root level of the project. Undo-able.  AEGP\_CreateComp(  AEGP\_ItemH parent\_folderHO, const A\_UTF16Char  \*utf\_nameZ,  A\_Long widthL,  A\_Long heightL, const A\_Ratio  \*pixel\_aspect\_ratioPRt, const A\_Time \*durationPT,  const A\_Ratio  \*frameratePRt, AEGP\_CompH \*new\_compPH); |
| AEGP\_GetNewCollectionFromComp Selection | Creates a new AEGP\_Collection2H from the items  selected in the given composition. The plug-in is responsible for disposing of the AEGP\_Collection2H.  AEGP\_GetNewCollectionFromCompSelection( AEGP\_PluginID plugin\_id,  AEGP\_CompH compH, AEGP\_Collection2H \*collectionPH); |
| AEGP\_SetSelection | Sets the selection within the given composition to the given AEGP\_Collection2H. Will return an error if members of the AEGP\_Collection2H are not available. Don’t assume that a composition hasn’t changed between  operations; always use a fresh AEGP\_Collection2H.  AEGP\_SetSelection(  AEGP\_CompH compH, AEGP\_Collection2H collectionH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompDisplayStartTime | Gets the displayed start time of a composition.  AEGP\_GetCompDisplayStartTime( AEGP\_CompH compH,  const A\_Time \*start\_timePT); |
| AEGP\_SetCompDisplayStartTime | Not undo-able. Sets the displayed start time of a composition (has no effect on the duration of the composition).  AEGP\_SetCompDisplayStartTime( AEGP\_CompH compH,  const A\_Time \*start\_timePT); |
| AEGP\_SetCompDuration | Undoable. Sets the duration of the given composition.  AEGP\_SetCompDuration( AEGP\_CompH compH,  const A\_Time \*durationPT); |
| AEGP\_CreateNullInComp | Creates a “null object” in the composition (useful for  translating projects from 3D applications into After Effects).  If you pass NULL for the duration, After Effects uses its  preference for the duration of a new still. If you pass 0, or an invalid time scale, duration is set to the length of the  composition.  AEGP\_CreateNullInComp(  const A\_UTF16Char \*utf\_nameZ, AEGP\_CompH parent\_compH, const A\_Time \*durationPT0, AEGP\_LayerH \*new\_null\_solidPH); |
| AEGP\_SetCompPixelAspectRatio | Sets the pixel aspect ratio of a composition.  AEGP\_SetCompPixelAspectRatio( AEGP\_CompH compH, const A\_Ratio \*parPRt); |
| AEGP\_CreateTextLayerInComp | Updated in CS6. Creates a text layer in the composition, and returns its AEGP\_LayerH.  AEGP\_CreateTextLayerInComp( AEGP\_CompH parent\_compH,  A\_Boolean select\_new\_layerB, AEGP\_LayerH \*new\_text\_lyrPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_CreateBoxTextLayerInComp | Updated in CS6. Creates a new box text layer, and returns its AEGP\_LayerH.  AEGP\_CreateBoxTextLayerInComp)( AEGP\_CompH parent\_compH, A\_Boolean select\_new\_layerB, A\_FloatPoint box\_dimensions, AEGP\_LayerH \*new\_text\_layerPH); |
| AEGP\_SetCompDimensions | Sets the dimensions of the composition. Undoable.  AEGP\_SetCompDimensions( AEGP\_CompH compH,  A\_long widthL,  A\_long heightL); |
| AEGP\_DuplicateComp | Duplicates the composition. Undoable.  AEGP\_DuplicateComp( AEGP\_CompH compH,  AEGP\_CompH \*new\_compPH); |
| AEGP\_GetCompFrameDuration | Retrieves the duration of a frame in a composition.  AEGP\_GetCompFrameDuration( AEGP\_CompH compH,  A\_Time \*timeP); |
| AEGP\_GetMostRecentlyUsedComp | Returns the most-recently-used composition.  AEGP\_GetMostRecentlyUsedComp( AEGP\_CompH \*compPH); |
| AEGP\_CreateVectorLayerInComp | Creates and returns a handle to a new vector layer.  AEGP\_CreateVectorLayerInComp( AEGP\_CompH parent\_compH, AEGP\_LayerH \*new\_vec\_layerPH); |
| AEGP\_GetNewCompMarkerStream | Returns an AEGP\_StreamRefH to the composition’s marker stream. Must be disposed by caller.  AEGP\_GetNewCompMarkerStream( AEGP\_PluginID aegp\_plugin\_id, AEGP\_CompH parent\_compH, AEGP\_StreamRefH \*streamPH); |
| AEGP\_GetCompDisplayDropFrame | Passes back a boolean that indicates whether the specified comp uses drop-frame timecode or not.  AEGP\_GetCompDisplayDropFrame( AEGP\_CompH compH,  A\_Boolean \*dropFramePB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetCompDisplayDropFrame | Sets the dropness of the timecode in the specified composition.  AEGP\_SetCompDisplayDropFrame( AEGP\_CompH compH,  A\_Boolean dropFrameB); |

**WORK WITH FOOTAGE**

Provides information about footage, or items in a project or composition. When getting and setting footage’s interpretation, it is possible to specify incompatible options. If you

encounter warnings and errors during development, be sure to make all related changes

atomically, and reassess the logic of the operation you’re performing. For example, changing the pull-down interpretation of footage won’t work unless there’s a difference between it’s

native and conformed frame rate.Depending on what you’re trying to accomplish, it may make sense to abort all of your operations at that point, inform the user of the problem encountered.

**TABLE 73: AEGP\_FOOTAGESUITE5**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetMainFootageFromItem | Returns an error if item isn’t a footage item Used to convert an item handle to a footage handle.  AEGP\_GetMainFootageFromItem( AEGP\_ItemH itemH, AEGP\_FootageH \*footagePH); |
| AEGP\_GetProxyFootageFromItem | Returns an error if item has no proxy. Returns the proxy footage handle. Note: a composition can have a proxy.  AEGP\_GetProxyFootageFromItem( AEGP\_ItemH itemH, AEGP\_FootageH \*proxy\_ftgPH); |
| AEGP\_GetFootageNumFiles | Returns the number of data (RGBA or audio) files, and the number of files per frame (may be greater than one if the footage has auxiliary channels).  AEGP\_GetFootageNumFiles( AEGP\_FootageH footageH, A\_long \*num\_filesPL0,  A\_long \*files\_per\_frmPL0); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetFootagePath | Get fully realized path to footage source file. Retrieves the footage path for a piece of footage (or for the specified frame of a footage sequence). frame\_numL  ranges from 0 to num\_main\_files, as obtained using  [AEGP\_GetFootageNumFiles](#_bookmark583). AEGP\_FOOTAGE\_MAIN\_FILE\_INDEX is the main file. The path is a handle to a NULL-terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetFootagePath( AEGP\_FootageH footageH, A\_long frame\_numL,  A\_long file\_indexL, AEGP\_MemHandle \*unicode\_pathPH); |
| AEGP\_GetFootageSignature | Retrieves the footage signature of specified footage.  AEGP\_GetFootageSignature( AEGP\_FootageH footageH, AEGP\_FootageSignature \*sigP);  The signature will be one of the following:  AEGP\_FootageSignature\_NONE AEGP\_FootageSignature\_MISSING AEGP\_FootageSignature\_SOLID |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewFootage | Creates a new footage item. The file path is a NULL-  terminated UTF-16 string with platform separators. Note that footage filenames with colons are not allowed, since colons are used as path separators in the HFS+ file system.  AEGP\_NewFootage(  AEGP\_PluginID aegp\_plugin\_id, const A\_UTF16Char  \*pathZ, const AEGP\_FootageLayerKey  \*layer\_infoP0, const AEGP\_FileSequenceImportOptions  \*sequence\_optionsP0,  AEGP\_InterpretationStyle  interp\_style,  void \*reserved, AEGP\_FootageH \*footagePH);  Note the optional params. If allow\_interpretation\_dialogB is FALSE, After Effects will guess the alpha interpretation.  typedef struct {  A\_long layer\_idL;  A\_long layer\_indexL  char \*nameAC;  AEGP\_LayerDrawStyle draw\_style;  } AEGP\_FootageLayerKey; AEGP\_LayerDrawStyle can be:  AEGP\_LayerDrawStyle\_LAYER\_BOUNDS AEGP\_LayerDrawStyle\_DOCUMENT\_BOUNDS  AEGP\_InterpretationStyle can be: AEGP\_InterpretationStyle\_NO\_DIALOG\_GUESS Will guess alpha interpretation even if file contains unknown alpha interpretation and user pref says to ask user.  AEGP\_InterpretationStyle\_DIALOG\_OK  Optionally can show a dialog.  AEGP\_InterpretationStyle\_NO\_DIALOG\_NO\_GUESS  Used for replace footage implementation. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_AddFootageToProject | Adds a footage item to a project. Footage will be adopted by the project, and may be added only once. This is  Undo-able; do not dispose of the returned added item if it’s undone.  AEGP\_AddFootageToProject( AEGP\_FootageH footageH, AEGP\_ItemH folderH,  AEGP\_ItemH \*add\_itemPH0); |
| AEGP\_SetItemProxyFootage | Sets footage as the proxy for an item. Will be adopted by the project. This is Undo-able; do not dispose of the  returned added item if it’s undone.  AEGP\_SetItemProxyFootage( AEGP\_FootageH footageH, AEGP\_ItemH itemH); |
| AEGP\_ReplaceItemMainFootage | Replaces footage for an item. The item will replace the main footage for this item. This is Undo-able; do not dispose of the returned added item if it’s undone.  AEGP\_ReplaceItemMainFootage( AEGP\_FootageH footageH,  AEGP\_ItemH itemH); |
| AEGP\_DisposeFootage | Deletes a footage item. Do not dispose of footage you did not create, or that has been added to the project.  AEGP\_DisposeFootage( AEGP\_FootageHfootageH); |
| AEGP\_GetFootageInterpretation | Populates an AEGP\_FootageInterp describing the settings of the AEGP\_FootageH. There is no way to create a valid AEGP\_FootageInterp other than by using this function.  AEGP\_GetFootageInterpretation( const AEGP\_ItemH itemH, A\_Boolean proxyB, AEGP\_FootageInterp \*interpP);  If proxyB is TRUE, the proxy footage’s settings are retrieved. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetFootageInterpretation | Apply the settings in the AEGP\_FootageInterp to the AEGP\_FootageH. Undo-able.  AEGP\_SetFootageInterpretation( const AEGP\_ItemH itemH, A\_Boolean proxyB, const AEGP\_FootageInterp  \*interpP);  If proxyB is TRUE, the proxy footage’s settings are modified. |
| AEGP\_GetFootageLayerKey | Populates an AEGP\_FootageLayerKey describing the footage.  AEGP\_GetFootageLayerKey( AEGP\_FootageH footageH, AEGP\_FootageLayerKey\*layerKeyP); |
| AEGP\_NewPlaceholderFootage | Deprecated. Adds a new placeholder footage item to the project. Using this function for missing footage will cause the user to search for each individual missing file, regardless of whether or not they’re all in the same  directory. Undo-able.  AEGP\_NewPlaceholderFootage( AEGP\_PluginID plugin\_id, const A\_char \*nameZ,  A\_long width,  A\_long height,  const A\_Time \*durationPT, AEGP\_FootageH \*footagePH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewPlaceholderFootage WithPath | This is the hip new way to add references to footage that can’t be found right this moment. The file path is a NULL-terminated UTF-16 string with platform  separators.  In CS6 and earlier, file\_type was ignored and we previously recommendedsetting it to AEIO\_FileType\_NONE. Starting in CC,  AEIO\_FileType\_NONE is now a warning condition. If you pass AEIO\_FileType\_ANY, then path MUST exist. If the path may not exist, pass AEIO\_FileType\_DIR for folder, or AEIO\_FileType\_GENERIC for a file.  AEGP\_NewPlaceholderFootageWithPath( AEGP\_PluginID plugin\_id, const A\_UTF16Char \*pathZ, AEGP\_Platform path\_platform, AEIO\_FileType file\_type, A\_long widthL,  A\_long heightL,  const A\_Time \*durationPT, AEGP\_FootageH \*footagePH); |
| AEGP\_NewSolidFootage | This is the way to add a solid. Until the footage is added to the project, the caller owns the AEGP\_FootageH (and must dispose of it if, and only if, it isn’t added to the project).  AEGP\_NewSolidFootage(  const A\_char \*nameZ,  A\_long width,  A\_long height,  const AEGP\_ColorVal \*colorP, AEGP\_FootageH \*footagePH); |
| AEGP\_GetSolidFootageColor | Returns the color of a given solid. Returns an error if the  AEGP\_ItemH is not a solid.  AEGP\_GetSolidFootageColor( AEGP\_ItemH itemH,  A\_Boolean proxyB, AEGP\_ColorVal \*colorP);  If proxyB is TRUE, the proxy solid’s color is retrieved. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetSolidFootageColor | Sets the color of a solid. Undo-able.  AEGP\_SetSolidFootageColor( AEGP\_ItemH itemH,  A\_Boolean proxyB,  AEGP\_ColorVal \*colorP);  If proxyB is TRUE, the proxy solid’s color is set. |
| AEGP\_SetSolidFootageDimensions | Sets the dimensions of a solid. Undo-able.  AEGP\_SetSolidFootageDimensions( AEGP\_ItemH itemH,  A\_Boolean proxyB,  A\_long widthL,  A\_long heightL);  If proxyB is TRUE, the proxy solid’s dimensions are modified. Returns an error if the item isn’t a solid. |
| AEGP\_GetFootageSoundDataFormat | Retrieves information about the audio data in the footage item (by populating the AEGP\_SoundDataFormat you passed in).  AEGP\_GetFootageSoundDataFormat( AEGP\_FootageH footageH, AEGP\_SoundDataFormat \*formatP); |
| AEGP\_GetFootageSequence ImportOptions | Populates and returns a AEGP\_FileSequenceImportOptions describing the given AEGP\_FootageH.  AEGP\_GetFootageSequenceImportOptions( AEGP\_FootageH footageH, AEGP\_FileSequenceImportOptions  \*optionsP); |

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| AEGP\_InterlaceLabel il; | The interlace settings for the footage item.  A\_u\_long signature; // ‘FIEL’  A\_short version; FIEL\_Type type; FIEL\_Order order; A\_u\_long reserved;  FIEL\_Type is one of the following:  FIEL\_Type\_FRAME\_RENDERED FIEL\_Type\_INTERLACED FIEL\_Type\_HALF\_HEIGHT FIEL\_Type\_FIELD\_DOUBLED  FIEL\_Type\_FIELD\_DOUBLED means 60 full-sized field doubled frames per second.  FIEL\_Order is either FIEL\_Order\_UPPER\_FIRST or FIEL\_Order\_LOWER\_FIRST. |
| AEGP\_AlphaLabel al; | AEGP\_AlphaFlag flags; A\_u\_char redCu;  A\_u\_char greenCu;  A\_u\_char blueCu;  AEGP\_AlphaFlags is one or more of the following, OR’d together:  AEGP\_AlphaPremul AEGP\_AlphaInverted AEGP\_AlphaIgnore  If AEGP\_AlphaPremul is not set, straight alpha is assumed. AEGP\_AlphaInverted indicates that  higher values are transparent, instead of lower. |
| AEGP\_PulldownPhase pd; | Indicates the phase for use in 3:2 pulldown. One of the following:  AEGP\_PulldownPhase\_NO\_PULLDOWN, AEGP\_PulldownPhase\_WSSWW, AEGP\_PulldownPhase\_SSWWW, AEGP\_PulldownPhase\_SWWWS, AEGP\_PulldownPhase\_WWWSS, AEGP\_PulldownPhase\_WWSSW, AEGP\_PulldownPhase\_WWWSW, AEGP\_PulldownPhase\_WWSWW, AEGP\_PulldownPhase\_WSWWW, AEGP\_PulldownPhase\_SWWWW, AEGP\_PulldownPhase\_WWWWS |

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| AEGP\_LoopBehavior loop; | Indicates the number of times the footage should loop.  A\_long loops;  A\_long reserved; |
| A\_Ratio pix\_aspect\_ratio; | Expresses the pixel aspect ratio of the footage (x over y). |
| A\_FpLong native\_fpsF; | The original framerate (in frames per second) of the footage item. |
| A\_FpLong conform\_fpsF; | The framerate being used for the footage item. |
| A\_long depthL; | The pixel depth of the footage. One of the following:  AEGP\_Footage\_Depth\_1 AEGP\_Footage\_Depth\_2 AEGP\_Footage\_Depth\_4 AEGP\_Footage\_Depth\_8 AEGP\_Footage\_Depth\_16 AEGP\_Footage\_Depth\_24 AEGP\_Footage\_Depth\_30 AEGP\_Footage\_Depth\_32 AEGP\_Footage\_Depth\_GRAY\_2 AEGP\_Footage\_Depth\_GRAY\_4 AEGP\_Footage\_Depth\_GRAY\_8 AEGP\_Footage\_Depth\_48 AEGP\_Footage\_Depth\_64 AEGP\_Footage\_Depth\_GRAY\_16 |
| A\_Boolean motion\_dB; | Indicates whether motion de-interlacing is being applied to the footage item. |

**MANAGE LAYERS**

AEGP\_LayerSuite provides information about layers within a composition, and the

relationship(s) between the source and layer times. As most After Effects usage boils down to layer manipulation, this is among the largest function suites in our API.

**TABLE 75: AEGP\_LAYERSUITE8**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompNumLayers | Obtains the number of layers in a composition.  AEGP\_GetCompNumLayers( AEGP\_CompH compH,  A\_long \*num\_layersPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompLayerByIndex | Get a AEGP\_LayerH from a composition. Zero is the foremost layer.  AEGP\_GetCompLayerByIndex( AEGP\_CompH compH,  A\_long layer\_indexL,  AEGP\_LayerH \*layerPH); |
| AEGP\_GetActiveLayer | Get the active layer. If a Layer or effect controls palette is  active, the active layer is that associated with the front-most tab in the window. If a composition or timeline window is active, the active layer is the selected layer (if only one is  selected; otherwise NULL is returned).  AEGP\_GetActiveLayer(  AEGP\_LayerH \*layerPH); |
| AEGP\_GetLayerIndex | Get the index of the layer (0 is the topmost layer in the composition).  AEGP\_GetLayerIndex( AEGP\_LayerH layerH,  A\_long \*layer\_indexPL); |
| AEGP\_GetLayerSourceItem | Get the AEGP\_ItemH of the layer’s source item.  AEGP\_GetLayerSourceItem( AEGP\_LayerH layerH,  AEGP\_ItemH \*source\_itemPH); |
| AEGP\_GetLayerSourceItemID | Retrieves the ID of the given AEGP\_LayerH. This is useful when hunting for a specific layer’s ID in an AEGP\_StreamVal.  AEGP\_GetLayerSourceItemID( AEGP\_LayerH layerH,  A\_long \*source\_idPL); |
| AEGP\_GetLayerParentComp | Get the AEGP\_CompH of the composition containing the layer.  AEGP\_GetLayerParentComp( AEGP\_LayerH layerH,  AEGP\_CompH \*compPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLayerName | Get the name of a layer. Both utf\_layer\_namePH0 and utf\_source\_namePH0 point to null terminated UTF- 16 strings. They must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetLayerName( AEGP\_PluginID pluginID, AEGP\_LayerH layerH,  AEGP\_MemHandle \*utf\_layer\_namePH0, AEGP\_MemHandle \*utf\_source\_namePH0); |
| AEGP\_GetLayerQuality | Get the quality of a layer.  AEGP\_GetLayerQuality( AEGP\_LayerH layerH,  AEGP\_LayerQuality \*qualityP);  Layer quality is one of the following flags:  AEGP\_LayerQual\_NONE AEGP\_LayerQual\_WIREFRAME AEGP\_LayerQual\_DRAFT AEGP\_LayerQual\_BEST |
| AEGP\_SetLayerQuality | Sets the quality of a layer (see flag values above). Undoable.  AEGP\_SetLayerQuality( AEGP\_LayerH layerH, AEGP\_LayerQuality quality); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLayerFlags | Get flags for a layer.  AEGP\_GetLayerFlags( AEGP\_LayerH layerH,  AEGP\_LayerFlags \*layer\_flagsP);  AEGP\_LayerFlag\_NONE AEGP\_LayerFlag\_VIDEO\_ACTIVE AEGP\_LayerFlag\_AUDIO\_ACTIVE AEGP\_LayerFlag\_EFFECTS\_ACTIVE AEGP\_LayerFlag\_MOTION\_BLUR AEGP\_LayerFlag\_FRAME\_BLENDING AEGP\_LayerFlag\_LOCKED AEGP\_LayerFlag\_SHY AEGP\_LayerFlag\_COLLAPSE AEGP\_LayerFlag\_AUTO\_ORIENT\_ROTATION AEGP\_LayerFlag\_ADJUSTMENT\_LAYER AEGP\_LayerFlag\_TIME\_REMAPPING AEGP\_LayerFlag\_LAYER\_IS\_3D AEGP\_LayerFlag\_LOOK\_AT\_CAMERA AEGP\_LayerFlag\_LOOK\_AT\_POI AEGP\_LayerFlag\_SOLO AEGP\_LayerFlag\_MARKERS\_LOCKED, AEGP\_LayerFlag\_NULL\_LAYER, AEGP\_LayerFlag\_HIDE\_LOCKED\_MASKS, AEGP\_LayerFlag\_GUIDE\_LAYER, AEGP\_LayerFlag\_ENVIRONMENT\_LAYER,  AEGP\_LayerFlag\_ADVANCED\_FRAME\_BLENDING,  True only if pixel motion frame blending is on for the layer.  AEGP\_LayerFlag\_SUBLAYERS\_RENDER\_SEPARATELY,  Used to get/set the state of per-character 3D enablement on a text layer.  AEGP\_LayerFlag\_ENVIRONMENT\_LAYER  New in CS6. |
| AEGP\_SetLayerFlag | Sets one layer flag at a time. Undoable.  AEGP\_SetLayerFlag(  AEGP\_LayerH layerH, AEGP\_LayerFlags single\_flag, A\_Boolean valueB); |
| AEGP\_IsLayerVideoReallyOn | Determines whether the layer’s video is visible. This is necessary to account for ‘solo’ status of other layers in the composition; non-solo’d layers are still on.  AEGP\_IsLayerVideoReallyOn( AEGP\_LayerH layerH,  A\_Boolean \*onPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_IsLayerAudioReallyOn | Accounts for solo status of other layers in the composition.  AEGP\_IsLayerAudioReallyOn( AEGP\_LayerH layerH,  A\_Boolean \*onPB); |
| AEGP\_GetLayerCurrentTime | Get current time, in layer or composition timespace. This value is not updated during rendering. NOTE: If a layer  starts at other than time 0 or is time-stretched other than 100%, layer time and composition time are distinct.  AEGP\_GetLayerCurrentTime( AEGP\_LayerH layerH, AEGP\_LTimeMode time\_mode,  A\_Time \*curr\_timePT); |
| AEGP\_GetLayerInPoint | Get time of first visible frame in composition or layer time. In layer time, the in\_pointPT is always 0.  AEGP\_GetLayerInPoint( AEGP\_LayerH layerH,  AEGP\_LTimeMode time\_mode,  A\_Time \*in\_pointPT); |
| AEGP\_GetLayerDuration | Get duration of layer, in composition or layer time, in seconds.  AEGP\_GetLayerDuration( AEGP\_LayerH layerH,  AEGP\_LTimeMode time\_mode,  A\_Time \*durationPT); |
| AEGP\_SetLayerInPointAnd Duration | Set duration and in point of layer in composition or layer time. Undo-able.  AEGP\_SetLayerInPointAndDuration( AEGP\_LayerH layerH, AEGP\_LTimeMode time\_mode, const A\_Time \*in\_pointPT,  const A\_Time \*durationPT); |
| AEGP\_GetLayerOffset | Get the offset from the start of the composition to layer time 0, in composition time.  AEGP\_GetLayerOffset( AEGP\_LayerH layerH,  A\_Time \*offsetPT); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetLayerOffset | Set the offset from the start of the composition to the first frame of the layer, in composition time. Undoable.  AEGP\_SetLayerOffset( AEGP\_LayerH layerH,  A\_Time \*offsetPT); |
| AEGP\_GetLayerStretch | Get stretch factor of a layer.  AEGP\_GetLayerStretch( AEGP\_LayerH layerH,  A\_Ratio \*stretchPRt); |
| AEGP\_SetLayerStretch | Set stretch factor of a layer.  AEGP\_SetLayerStretch( AEGP\_LayerH layerH,  A\_Ratio \*stretchPRt); |
| AEGP\_GetLayerTransferMode | Get transfer mode of a layer.  AEGP\_GetLayerTransferMode( AEGP\_LayerH layerH, AEGP\_LayerTransferMode \*modeP); |
| AEGP\_SetLayerTransferMode | Set transfer mode of a layer. Undoable.  AEGPSetLayerTransferMode( AEGP\_LayerH layerH, AEGP\_LayerTransferMode \*modeP);  As of 6.5, when you make a layer a track matte, the layer in front of it will be disabled, as when you do this via the  interface. |
| AEGP\_IsAddLayerValid | Tests whether it’s currently valid to add a given item to a composition. A composition cannot be added to itself, or to any compositions which it contains; other conditions can  preclude successful adding too. Adding a layer without first using this function will produce undefined results.  AEGP\_IsAddLayerValid(  AEGP\_ItemH item\_to\_addH,  AEGP\_CompH into\_compH,  A\_Boolean \*validPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_AddLayer | Add an item to the composition, above all other layers. Undo-able. Use [AEGP\_IsAddLayerValid()](#_bookmark600) first, to confirm that it’s possible.  AEGP\_AddLayer(  AEGP\_ItemH item\_to\_addH,  AEGP\_CompH into\_compH,  A\_Boolean \*added\_layerPH0); |
| AEGP\_ReorderLayer | Change the order of layers. Undoable.  AEGP\_ReorderLayer( AEGP\_LayerH layerH,  A\_long layer\_indexL);  To add a layer to the end of the composition, to use layer\_indexL = AEGP\_REORDER\_LAYER\_TO\_END |
| AEGP\_GetLayerMaskedBounds | Given a layer’s handle and a time, returns the bounds of area visible with masks applied.  AEGP\_GetLayerMaskedBounds( AEGP\_LayerH layerH,  const A\_Time \*comp\_timePT,  A\_FloatRect \*boundsPR); |
| AEGP\_GetLayerObjectType | Returns a layer’s object type.  AEGP\_GetLayerObjectType( AEGP\_LayerH layerH,  AEGP\_ObjectType \*object\_type);  AEGP\_ObjectType\_AV AEGP\_ObjectType\_LIGHT AEGP\_ObjectType\_CAMERA, AEGP\_ObjectType\_TEXT |
| AEGP\_IsLayer3D | Is the footage item a 3D layer. All AV layers are either 2D or 3D.  AEGP\_IsLayer3D(  AEGP\_LayerH layerH,  A\_Boolean \*is\_3DPB); |
| AEGP\_IsLayer2D | Is the footage item a 2D layer. All AV layers are either 2D or 3D.  AEGP\_IsLayer2D(  AEGP\_LayerH layerH,  A\_Boolean \*is\_2DPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_IsVideoActive | Given composition time and a layer, see if the layer will render. Time mode is either AEGP\_LTimeMode\_LayerTime or AEGP\_LTimeMode\_CompTime.  AEGP\_IsVideoActive( AEGP\_LayerH layerH,  AEGP\_LTimeMode time\_mode, A\_Time \*comp\_timePT,  A\_Boolean \*is\_activePB); |
| AEGP\_IsLayerUsedAsTrackMatte | Is the layer used as a track matte?  AEGP\_IsLayerUsedAsTrackMatte( AEGP\_LayerH layerH,  A\_Boolean fill\_must\_be\_activeB,  A\_Boolean \*is\_track\_mattePB); |
| AEGP\_DoesLayerHaveTrackMatte | Does this layer have a Track Matte?  AEGP\_DoesLayerHaveTrackMatte( AEGP\_LayerH layerH,  A\_Boolean \*has\_track\_mattePB); |
| AEGP\_ConvertCompToLayerTime | Given a time in composition space, returns the time relative to the layer source footage.  AEGP\_ConvertCompToLayerTime( AEGP\_LayerH layerH,  const A\_Time \*comp\_timeP,  A\_Time \*layer\_timeP); |
| AEGP\_ConvertLayerToCompTime | Given a time in layer space, find the corresponding time in composition space.  AEGP\_ConvertLayerToCompTime( AEGP\_LayerH layerH,  const A\_Time \*layer\_timePT,  A\_Time \*comp\_timePT); |
| AEGP\_GetLayerDancingRandValue | Used by the dancing dissolve transfer function.  AEGP\_GetLayerDancingRandValue( AEGP\_LayerH layerH,  const A\_Time \*comp\_timePT,  A\_long \*rand\_valuePL); |
| AEGP\_GetLayerID | Supplies the layer’s unique ID. This ID never changes during the lifetime of the project.  AEGP\_GetLayerID(  AEGP\_LayerH layerH, AEGP\_LayerIDVal \*id\_valP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLayerToWorldXform | Given a layer handle and time, returns the layer-to-world transformation matrix.  AEGP\_GetLayerToWorldXform( AEGP\_LayerH aegp\_layerH,  const A\_Time \*comp\_timeP,  A\_Matrix4 \*transform); |
| AEGP\_GetLayerToWorldXformFrom View | Given a layer handle, the current (composition) time, and the requested view time, returns the translation between the user’s view and the layer, corrected for the composition’s  current aspect ratio.  AEGP\_GetLayerToWorldXformFromView( AEGP\_LayerH aegp\_layerH,  const A\_Time \*view\_timeP,  const A\_Time \*comp\_timeP,  A\_Matrix4 \*transform); |
| AEGP\_SetLayerName | Sets the name of a layer. Undo-able. new\_nameZ points to a null terminated UTF-16 string.  AEGP\_SetLayerName(  AEGP\_LayerH aegp\_layerH, const A\_UTF16Char \*new\_nameZ); |
| AEGP\_GetLayerParent | Retrieves the handle to a layer’s parent (none if not parented).  AEGP\_GetLayerParent( AEGP\_LayerH layerH,  AEGP\_LayerH \*parent\_layerPH); |
| AEGP\_SetLayerParent | Sets a layer’s parent layer.  AEGP\_SetLayerParent( AEGP\_LayerH layerH,  const AEGP\_LayerH parent\_layerH); |
| AEGP\_DeleteLayer | Deletes a layer. Can you believe it took us three suite versions to add a delete function? Neither can we.  AEGP\_DeleteLayer(  AEGP\_LayerH layerH); |
| AEGP\_DuplicateLayer | Duplicates the layer. Undoable.  AEGP\_DuplicateLayer(  AEGP\_LayerH orig\_layerH,  AEGP\_LayerH \*dupe\_layerPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLayerFromLayerID | Retrieves the AEGP\_LayerH associated with a given  AEGP\_LayerIDVal (which is what you get when accessing an effect’s layer parameter stream).  AEGP\_GetLayerFromLayerID( AEGP\_CompH parent\_compH,  AEGP\_LayerIDVal id,  AEGP\_LayerH \*layerPH); |
| AEGP\_GetLayerLabel | Gets a layer’s AEGP\_LabelID.  AEGP\_GetLayerLabel( AEGP\_LayerH layerH,  AEGP\_LabelID \*labelP); |
| AEGP\_SetLayerLabel | Sets a layer’s AEGP\_LabelID. Undoable.  AEGP\_SetLayerLabel( AEGP\_LayerH layerH,  AEGP\_LabelID label); |
| AEGP\_GetLayerSamplingQuality | New in CC. Get the sampling quality of a layer.  AEGP\_GetLayerSamplingQuality( AEGP\_LayerH layerH, AEGP\_LayerSamplingQuality \*label);  Layer sampling quality is one of the following flags:  AEGP\_LayerSamplingQual\_BILINEAR AEGP\_LayerSamplingQual\_BICUBIC |
| AEGP\_SetLayerSamplingQuality | New in CC. Sets the sampling quality of a layer (see flag values above). Option is explicitly set on the layer  independent of layer quality. If you want to force it on you must also set the layer quality to AEGP\_LayerQual\_BEST with [AEGP\_SetLayerQuality](#_bookmark595). Otherwise it will only be using the specified layer sampling quality whenever the  layer quality is set to AEGP\_LayerQual\_BEST. Undoable.  AEGP\_SetLayerSamplingQuality( AEGP\_LayerH layerH, AEGP\_LayerSamplingQuality label); |

**LAYER CREATION NOTES**

All layers created using AEGP calls will start at composition time 0, and have the duration of the composition. Use [AEGP\_SetLayerOffset](#_bookmark598)() and [AEGP\_SetLayerInPointAndDuration](#_bookmark597)() to properly set the layer’s time information.

When the layer stretch factor (obtained using [AEGP\_GetLayerStretch](#_bookmark599), naturally) is not 100%, the following computation will be needed to yield the correct layer offset:

offset = compIn - stretch \* layerIn;

**COMMUNICATION WITH A LAYER’S EFFECTS**

Access the effects applied to a layer. This suite provides access to all parameter data streams. Use the [AEGP\_StreamSuite](#_bookmark622) to work with those streams.

An AEGP\_Effect\_RefH is a reference to an applied effect. An AEGP\_InstalledEffectKey is a reference to an installed effect, which may or may not be currently applied to anything. If Foobarocity is applied to a layer twice, there will be two

distinct AEGP\_Effect\_RefHs, but they’ll both return the same

AEGP\_InstalledEffectKey.

**TABLE 76: AEGP\_EFFECTSUITE4**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLayerNumEffects | Get number of effects applied to a layer.  AEGP\_GetLayerNumEffects( AEGP\_LayerH layerH,  A\_long \*num\_effectsPL); |
| AEGP\_GetLayerEffectByIndex | Retrieves (by index) a reference to an effect applied to the layer.  AEGP\_GetLayerEffectByIndex( AEGP\_PluginID aegp\_plugin\_id, AEGP\_LayerH layerH, AEGP\_EffectIndex effect\_indexL, AEGP\_EffectRefH \*effectPH); |
| AEGP\_GetInstalledKeyFromLayerE ffect | Given an AEGP\_EffectRefH, retrieves its associated  AEGP\_InstalledEffectKey.  AEGP\_GetInstalledKeyFromLayerEffect( AEGP\_EffectRefH effect\_refH, AEGP\_InstalledEffectKey  \*installed\_keyP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetEffectParamUnionByInde x | Returns description of effect parameter. Do not use the value(s) in the ParamDef returned by this function (Use  [AEGP\_GetNewStreamValue()](#_bookmark629) instead); it’s provided so AEGPs can access parameter defaults, checkbox names, and pop-up strings.  Use [AEGP\_GetEffectNumParamStreams()](#_bookmark625) from the StreamSuite to get the stream count, useful for  determining the maximum param\_index. The last parameter is optional;  AEGP\_GetEffectParamUnionByIndex( AEGP\_PluginID aegp\_plugin\_id, AEGP\_EffectRefH effectH, PF\_ParamIndex param\_index,  PF\_ParamType \*param\_typeP PF\_ParamDefUnion \*uP0); |
| AEGP\_GetEffectFlags | Obtains the flags for the given AEGP\_EffectRefH.  AEGP\_GetEffectFlags( AEGP\_EffectRefH effect\_refH,  AEGP\_EffectFlags \*effect\_flagsP);  Flags will be a combination of the following:  AEGP\_EffectFlags\_NONE AEGP\_EffectFlags\_ACTIVE AEGP\_EffectFlags\_AUDIO\_ONLY AEGP\_EffectFlags\_AUDIO\_TOO AEGP\_EffectFlags\_MISSING |
| AEGP\_SetEffectFlags | Sets the flags (enumerated above) for the given AEGP\_EffectRefH, masked by a different set of effect flags.  AEGP\_SetEffectFlags( AEGP\_EffectRefH effect\_refH, AEGP\_EffectFlags mask,  AEGP\_EffectFlags effect\_flags); |
| AEGP\_ReorderEffect | Change the order of applied effects (pass the requested index).  AEGP\_ReorderEffect( AEGP\_EffectRefH effect\_refH,  A\_long effect\_indexL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_EffectCallGeneric | Call an effect plug-in, and pass it a pointer to any data you like; the effect can modify it. This is how AEGPs  communicate with effects. Pass  PF\_Cmd\_COMPLETELY\_GENERAL for effect\_cmd.  AEGP\_EffectCallGeneric(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_EffectRefH effectH,  const A\_Time \*timePT,  PF\_Cmd effect\_cmd,  void \*extraPV); |
| AEGP\_DisposeEffect | Disposes of an AEGP\_EffectRefH. Use this to dispose of any AEGP\_EffectRefH returned by After Effects.  AEGP\_DisposeEffect( AEGP\_EffectRefH effectH); |
| AEGP\_ApplyEffect | Apply an effect to a given layer. Returns the newly-created  AEGP\_EffectRefH.  AEGP\_ApplyEffect(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_LayerH layerH,  AEGP\_InstalledEffectKey  installed\_key, AEGP\_EffectRefH \*effect\_refPH); |
| AEGP\_DeleteLayerEffect | Remove an applied effect.  AEGP\_DeleteLayerEffect( AEGP\_EffectRefH effect\_refH); |
| AEGP\_GetNumInstalledEffects | Returns the count of effects installed in After Effects.  AEGP\_GetNumInstalledEffects(  A\_long \*num\_installed\_effectsPL); |
| AEGP\_GetNextInstalledEffect | Returns the AEGP\_InstalledEffectKey of the next installed effect. Pass AEGP\_InstalledEffectKey\_NONE as the first  parameter to obtain the first  AEGP\_InstalledEffectKey.  AEGP\_GetNextInstalledEffect( AEGP\_InstalledEffectKey key, AEGP\_InstalledEffectKey  \*next\_keyPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetEffectName | Get name of the effect. nameZ can be up to  AEGP\_MAX\_EFFECT\_NAME\_SIZE + 1 long.  AEGP\_GetEffectName(  AEGP\_InstalledEffectKey  installed\_key,  A\_char \*nameZ);  Note: use [AEGP\_SetStreamName](#_bookmark638) to change the display name of an effect. |
| AEGP\_GetEffectMatchName | Get match name of an effect (defined in PiPL). match\_nameZ up to AEGP\_MAX\_EFFECT\_MATCH\_NAME\_SIZE + 1 long.  AEGP\_GetEffectMatchName(  AEGP\_InstalledEffectKey  installed\_key,  A\_char \*match\_nameZ);  Match names are in 7-bit ASCII. UI names are in the current application runtime encoding; for example, ISO 8859-1 for most languages on Windows. |
| AEGP\_GetEffectCategory | Menu category of effect. categoryZ can be up to  AEGP\_MAX\_EFFECT\_CATEGORY\_NAME\_SIZE + 1  long.  AEGP\_GetEffectCategory(  AEGP\_InstalledEffectKey  installed\_key,  A\_char \*categoryZ); |
| AEGP\_DuplicateEffect | Duplicates a given AEGP\_EffectRefH. Caller must dispose of duplicate when finished.  AEGP\_DuplicateEffect(  AEGP\_EffectRefH orig\_effect\_refH, AEGP\_EffectRefH \*dupe\_refPH); |
| AEGP\_NumEffectMask | New in CC 2014. How many masks are on this effect?  AEGP\_NumEffectMask( AEGP\_EffectRefH effect\_refH,  A\_u\_long \*num\_masksPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetEffectMaskID | New in CC 2014. For a given mask\_indexL, returns the corresponding AEGP\_MaskIDVal for use in uniquely  identifying the mask.  AEGP\_GetEffectMaskID( AEGP\_EffectRefH effect\_refH, A\_u\_long mask\_indexL, AEGP\_MaskIDVal \*id\_valP); |
| AEGP\_AddEffectMask | New in CC 2014. Add an effect mask, which may be created using the [Mask Suite](#_bookmark653). Returns the local stream of the effect ref - useful if you want to add  keyframes. Caller must dispose of AEGP\_StreamRefH  when finished. Undoable.  AEGP\_AddEffectMask( AEGP\_EffectRefH effect\_refH, AEGP\_MaskIDVal id\_val, AEGP\_StreamRefH streamPH0); |
| AEGP\_RemoveEffectMask | New in CC 2014. Remove an effect mask. Undoable.  AEGP\_RemoveEffectMask( AEGP\_EffectRefH effect\_refH, AEGP\_MaskIDVal id\_val); |
| AEGP\_SetEffectMask | New in CC 2014. Set an effect mask on an existing index.  Returns the local stream of the effect ref - useful if you want to add keyframes. Caller must dispose of AEGP\_StreamRefH when finished. Undoable.  AEGP\_SetEffectMask( AEGP\_EffectRefH effect\_refH, A\_u\_long mask\_indexL, AEGP\_MaskIDVal id\_val, AEGP\_StreamRefH \*streamPH0); |

**EXPLOITING EFFECT UI BEHAVIOR TO LOOK COOL**

Even if you manipulate a layer’s effects, its effect controls won’t necessarily become visible. However, if you [apply](#_bookmark611) then immediately [remove](#_bookmark612) an effect, the layer’s effect controls will be made visible. Tricky, eh?

**STREAMREFS AND EFFECTREFS**

How do you get an AEGP\_StreamRef for an effect? Start by getting the effect’s

AEGP\_EffectRef, by calling AEGP\_GetNewEffectForEffect(). Then call

AEGP\_GetNewEffectStreamByIndex(), say for param index 1, which passes back a parameter stream. Then call AEGP\_GetNewParentStreamRef() - voila, your

AEGP\_StreamRef sir!

If you acquire references to an effect’s streams, do not dispose of the AEGP\_EffectRefH

until you’re done with the streams, or you’ll unbalance After Effects’ checkout mechanism.

Also remember that AEGP\_StreamRefHs are opaque; AEGP\_StreamValue2s are not (entirely).

To get an effect’s instance name (as renamed by the user), get the AEGP\_StreamRef for the effect itself and call [AEGP\_GetStreamName](#_bookmark628).

**DIVING INTO STREAMS!**

Just about everything in After Effects is a stream. Effect parameters, layers, masks, and shapes are all internally represented by streams. The AEGP API can access nearly every aspect of every stream.

The After Effects timeline can contain numerous object types; each object supports a set of parameters called streams. All streams, regardless of which type of object to which they’re attached, are conceptually similar (and handled similarly by After Effects. But the way you access each type of stream varies because of their containment.

A stream, once acquired, represents a value which may change over time. Not all streams *can*

vary over time, and a particular stream may not be time-variant at the time of access.

There are two ways to access the value of a stream. If the stream has keyframes, you can use the [keyframe suite](#_bookmark641). The values provided won’t reflect the influence of expressions. Note: In any expression, the current keyframed value is always available as the variable value.

You can also use [AEGP\_GetNewStreamValue](#_bookmark629), which samples the value of the stream at a particular time. For streams without expressions or keyframes, the time parameter is

meaningless, and the function returns what essentially is the constant value of the stream.

Use [AEGP\_SetStreamValue](#_bookmark630) (which doesn't take a time as a parameter) to set these streams.

Many StreamSuite functions populate a StreamH, which your AEGP must dispose. when done. After Effects allocates and passes you a copy of the values, not a direct handle to the original value. [AEGP\_GetNewLayerStream()](#_bookmark624) is restricted to streams for which no

memory allocation is required to access their values.

**OKAY, WHAT DID I JUST GET?**

A stream value is a large union, only one structure of which (depending on the stream type) is populated. Note the similarity to the [PF\_ParamDef](#_bookmark207).

typedef union {

AEGP\_FourDVal four\_d;

AEGP\_ThreeDVal three\_d;

AEGP\_TwoDVal two\_d;

AEGP\_OneDVal one\_d;

AEGP\_ColorVal color;

AEGP\_ArbBlockVal arbH;

AEGP\_MarkerValP markerP;

AEGP\_LayerIDVal layer\_id;

AEGP\_MaskIDVal mask\_id; AEGP\_MaskOutlineValH mask; AEGP\_TextDocumentH text\_documentH;

} AEGP\_StreamVal2;

**LAYERS**

[AEGP\_GetLayerStreamValue](#_bookmark631) is used to access the parameters like anchor point and

position, native to almost all layers in AE. Use IsStreamLegal to allow you to determine if that stream type is offered on that layer.

**MASKS**

Since a layer can have multiple masks, access the masks using [AEGP\_GetLayerMaskByIndex](#_bookmark655). Masks don’t have streams like layers do; they get their own enumeration. Access their streams using [AEGP\_GetNewMaskStream](#_bookmark627).

**EFFECTS**

They can have a variable number of streams/parameters, and the order and definition of them is not known when the AEGP is written. Therefore we cannot offer an enum for

selecting them, and instead you must get them by index, hence [GetNewEffectStream-](#_bookmark626) [ByIndex](#_bookmark626).

Access and manipulate the values of a layer’s streams. For paint and text streams, use [AEGP\_DynamicStreamSuite](#_bookmark633) instead.

**TABLE 77: AEGP\_STREAMSUITE4**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_IsStreamLegal | Determines if the given stream is appropriate for the given layer.  AEGP\_IsStreamLegal( AEGP\_LayerH layerH,  AEGP\_LayerStream which\_stream, A\_Boolean\* is\_legalP); |
| AEGP\_CanVaryOverTime | Given a stream, returns whether or not a stream is time- variant (and can be keyframed).  AEGP\_CanVaryOverTime( AEGP\_StreamRefH streamH, A\_Boolean \*can\_varyPB); |
| AEGP\_GetValidInterpolations | Retrieves an AEGP\_KeyInterpolationMask indicating which interpolation types are valid for the AEGP\_StreamRefH.  AEGP\_GetValidInterpolations( AEGP\_StreamRefH streamH, AEGP\_KeyInterpolationMask  \*valid\_interpP);  AEGP\_KeyInterpolationMask will be a combination of the following:  AEGP\_KeyInterpMask\_NONE AEGP\_KeyInterpMask\_LINEAR AEGP\_KeyInterpMask\_BEZIER AEGP\_KeyInterpMask\_HOLD AEGP\_KeyInterpMask\_CUSTOM AEGP\_KeyInterpMask\_ANY |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewLayerStream | Get a layer’s data stream. Plug-in must dispose of streamPH. Note that this will not provide keyframe access; Use the [AEGP\_KeyframeSuite](#_bookmark641) instead.  AEGP\_GetNewLayerStream( AEGP\_PluginID id,  AEGP\_LayerH layerH, AEGP\_LayerStream which\_stream, AEGP\_StreamRefH \*streamPH);  AEGP\_LayerStream\_ANCHORPOINT AEGP\_LayerStream\_POSITION AEGP\_LayerStream\_SCALE AEGP\_LayerStream\_ROTATION AEGP\_LayerStream\_ROTATE\_Z AEGP\_LayerStream\_OPACITY AEGP\_LayerStream\_AUDIO AEGP\_LayerStream\_MARKER AEGP\_LayerStream\_TIME\_REMAP AEGP\_LayerStream\_ROTATE\_X, AEGP\_LayerStream\_ROTATE\_Y, AEGP\_LayerStream\_ORIENTATION,  Only valid for AEGP\_ObjectType\_CAMERA: AEGP\_ObjectType\_CAMERA AEGP\_LayerStream\_ZOOM, AEGP\_LayerStream\_DEPTH\_OF\_FIELD, AEGP\_LayerStream\_FOCUS\_DISTANCE, AEGP\_LayerStream\_APERTURE, AEGP\_LayerStream\_BLUR\_LEVEL, AEGP\_LayerStream\_IRIS\_SHAPE, AEGP\_LayerStream\_IRIS\_ROTATION, AEGP\_LayerStream\_IRIS\_ROUNDNESS, AEGP\_LayerStream\_IRIS\_ASPECT\_RATIO, AEGP\_LayerStream\_IRIS\_DIFFRACTION\_FRINGE, AEGP\_LayerStream\_IRIS\_HIGHLIGHT\_GAIN, AEGP\_LayerStream\_IRIS\_HIGHLIGHT\_THRESHOLD, AEGP\_LayerStream\_IRIS\_HIGHLIGHT\_SATURATION,  Only valid for AEGP\_ObjectType\_LIGHT: AEGP\_LayerStream\_INTENSITY, AEGP\_LayerStream\_COLOR, AEGP\_LayerStream\_CONE\_ANGLE, AEGP\_LayerStream\_CONE\_FEATHER, AEGP\_LayerStream\_SHADOW\_DARKNESS, AEGP\_LayerStream\_SHADOW\_DIFFUSION, AEGP\_LayerStream\_LIGHT\_FALLOFF\_TYPE, AEGP\_LayerStream\_LIGHT\_FALLOFF\_START, AEGP\_LayerStream\_LIGHT\_FALLOFF\_DISTANCE, |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewLayerStream  (continued) | Only valid for AEGP\_ObjectType\_AV: AEGP\_LayerStream\_ACCEPTS\_SHADOWS, AEGP\_LayerStream\_ACCEPTS\_LIGHTS, AEGP\_LayerStream\_AMBIENT\_COEFF, AEGP\_LayerStream\_DIFFUSE\_COEFF, AEGP\_LayerStream\_SPECULAR\_INTENSITY, AEGP\_LayerStream\_SPECULAR\_SHININESS, AEGP\_LayerStream\_METAL, AEGP\_LayerStream\_LIGHT\_TRANSMISSION, Only valid for AEGP\_ObjectType\_AV, new in CS6: AEGP\_LayerStream\_REFLECTION\_INTENSITY, AEGP\_LayerStream\_REFLECTION\_SHARPNESS, AEGP\_LayerStream\_REFLECTION\_ROLLOFF, AEGP\_LayerStream\_TRANSPARENCY\_COEFF, AEGP\_LayerStream\_TRANSPARENCY\_ROLLOFF, AEGP\_LayerStream\_INDEX\_OF\_REFRACTION,  AEGP\_LayerStream\_EXTRUSION\_BEVEL\_STYLE, AEGP\_LayerStream\_EXTRUSION\_BEVEL\_DIRECTION, AEGP\_LayerStream\_EXTRUSION\_BEVEL\_DEPTH, AEGP\_LayerStream\_EXTRUSION\_HOLE\_BEVEL\_DEPTH, AEGP\_LayerStream\_EXTRUSION\_DEPTH, AEGP\_LayerStream\_PLANE\_CURVATURE, AEGP\_LayerStream\_PLANE\_SUBDIVISION,  Only valid for LIGHT and AV only:  AEGP\_LayerStream\_CASTS\_SHADOWS, AEGP\_LayerStream\_SOURCE\_TEXT  AEGP\_LayerStream\_BEGIN = AEGP\_LayerStream\_ANCHORPOINT,  AEGP\_LayerStream\_END =  AEGP\_LayerStream\_LIGHT\_FALLOFF\_DISTANCE+1  enum {  AEGP\_LightFalloff\_NONE = 0, AEGP\_LightFalloff\_SMOOTH, AEGP\_LightFalloff\_INVERSE\_SQUARE\_CLAMPED  };  typedef A\_u\_long AEGP\_LightFalloffType; |
| AEGP\_GetEffectNumParamStreams | Get number of parameter streams associated with an effect.  AEGP\_GetEffectNumParamStreams( AEGP\_EffectRefH effect\_refH, A\_long \*num\_parmsPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewEffectStream ByIndex | Get an effect’s parameter stream. Plug-in must dispose of  streamPH.  AEGP\_GetNewEffectStreamByIndex( AEGP\_PluginID id, AEGP\_EffectRefH effect\_refH, PF\_ParamIndex param\_index, AEGP\_StreamRefH \*streamPH); |
| AEGP\_GetNewMaskStream | Get a mask’s stream. The stream must be disposed. Also see the [AEGP\_MaskSuite](#_bookmark651) and [AEGP\_MaskOutlineSuite](#_bookmark657) for additional Mask  functions.  AEGP\_MaskStream\_OUTLINE, AEGP\_MaskStream\_OPACITY, AEGP\_MaskStream\_FEATHER, AEGP\_MaskStream\_EXPANSION,  Useful for iteration:  AEGP\_MaskStream\_BEGIN = AEGP\_MaskStream\_OUTLINE,  AEGP\_MaskStream\_END = AEGP\_MaskStream\_EXPANSION+1  AEGP\_GetNewMaskStream(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_MaskRefH mask\_refH, AEGP\_MaskStream which\_stream, AEGP\_StreamRefH \*mask\_strmPH); |
| AEGP\_DisposeStream | Dispose of a stream (do this with all streams passed to the plug-in by these functions).  AEGP\_DisposeStream( AEGP\_StreamRefH streamH); |
| AEGP\_GetNewMaskOpacity | Get the mask’s opacity stream. The stream must be disposed.  AEGP\_GetNewMaskOpacity(  AEGP\_PluginID aegp\_plugin\_id,  AEGP\_MaskH maskH,  PF\_ParamIndex param\_index, AEGP\_StreamRefH  \*mask\_opacity\_streamPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetStreamName | Get name of the stream (localized or forced English). is handle of A\_UTF16Char (contains null terminated  UTF16 string); must be disposed with  AEGP\_FreeMemHandle.  AEGP\_GetStreamName( AEGP\_PluginID pluginID,  AEGP\_StreamRefH  streamH, A\_Boolean force\_englishB, AEGP\_MemHandle  \*utf\_stream\_namePH);  NOTE: if force\_englishB is TRUE, the default name will override any stream renaming which has been done  (either programatically, or by the user). |
| AEGP\_GetStreamUnitsText | Get stream units, formatted as text (localized or forced English); unitsZ up to AEGP\_MAX\_STREAM\_NAME\_LEN + 1 long.  AEGP\_GetStreamUnitsText( AEGP\_StreamRefH streamH, A\_Boolean force\_englishB,  A\_char \*unitsZ); |
| AEGP\_GetStreamProperties | Get stream’s flags, as well as minimum and maximum values (as floats), if the stream *has* mins and maxes.  StreamFlags values:  AEGP\_StreamFlag\_NONE AEGP\_StreamFlag\_HAS\_MIN AEGP\_StreamFlag\_HAS\_MAX  AEGP\_GetStreamProperties( AEGP\_StreamRefH streamH, AEGP\_StreamFlags \*flagsP, A\_FpLong \*minP0,  A\_FpLong \*maxP0); |
| AEGP\_IsStreamTimevarying | Returns whether or not the stream is affected by expressions.  AEGP\_IsStreamTimevarying( AEGP\_StreamRefH streamH, A\_Boolean \*is\_timevaryPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetStreamType | Get type (dimension) of a stream.  AEGP\_GetStreamType( AEGP\_StreamRefH streamH,  AEGP\_StreamType \*stream\_typeP);  AEGP\_StreamType\_NO\_DATA, AEGP\_StreamType\_TwoD\_SPATIAL, AEGP\_StreamType\_TwoD, AEGP\_StreamType\_ThreeD, AEGP\_StreamType\_ThreeD\_SPATIAL, AEGP\_StreamType\_OneD, AEGP\_StreamType\_COLOR, AEGP\_StreamType\_ARB, AEGP\_StreamType\_MARKER, AEGP\_StreamType\_LAYER\_ID, AEGP\_StreamType\_MASK\_ID, AEGP\_StreamType\_MASK, AEGP\_StreamType\_TEXT\_DOCUMENT  NOTE: always returns ThreeD\_Spatial for position, regardless of whether or not the layer is 3D. |
| AEGP\_GetNewStreamValue | Get value, at a time you specify, of stream. valueP must be disposed by the plug-in. The AEGP\_LTimeMode  indicates whether the time is in compositions or layer time.  AEGP\_GetNewStreamValue( AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH, AEGP\_LTimeMode time\_mode, const A\_Time \*timePT,  A\_Boolean pre\_exprB, AEGP\_StreamValue2 \*valueP); |
| AEGP\_DisposeStreamValue | Dispose of stream value. Always deallocate values passed to the plug-in.  AEGP\_DisposeStreamValue( AEGP\_StreamValue2 \*valueP); |
| AEGP\_SetStreamValue | Only legal when stream is not time-variant.  AEGP\_SetStreamValue(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH, AEGP\_StreamValue2 \*valueP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLayerStreamValue | NOTE: This convenience function is only valid for streams with primitive data types, and not for AEGP\_ArbBlockVal, AEGP\_MarkerValH or AEGP\_MaskOutlineValH. For these and other  complex types, use  AEGP\_GetNewStreamValue,described above.  AEGP\_GetLayerStreamValue( AEGP\_LayerH layerH, AEGP\_LayerStream which\_stream, AEGP\_LTimeMode time\_mode, const A\_Time \*timePT,  A\_Boolean pre\_expB, AEGP\_StreamVal \*stream\_valP, AEGP\_StreamType \*strm\_typeP0); |
| AEGP\_GetExpressionState | Determines whether expressions are enabled on the given  AEGP\_StreamRefH.  AEGP\_GetExpressionState)( AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH, A\_Boolean \*enabledPB); |
| AEGP\_SetExpressionState | Enables and disables expressions on the given  AEGP\_StreamRefH.  AEGP\_SetExpressionState( AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH, A\_Boolean enabledB); |
| AEGP\_GetExpression | Obtains the expression’s text.  AEGP\_GetExpression(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH, AEGP\_MemHandle \*expressionHZ); |
| AEGP\_SetExpression | Sets the expression’s text.  AEGP\_SetExpression(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH,  const char\* expressionP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_DuplicateStreamRef | Duplicates a given AEGP\_StreamRefH. Dispose of the duplicate.  AEGP\_DuplicateStreamRef( AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH, AEGP\_StreamRefH \*dup\_streamPH); |

**DYNAMIC STREAMS**

AEGP\_DynamicStreamSuite accesses and manipulates paint and text streams. Use AEGP\_GetStreamGroupingType and AEGP\_GetDynamicStreamFlags to identify the stream before attempting to use functions which only work on certain stream types. Also note that, often, you can simply use [AEGP\_StreamSuite](#_bookmark622) calls to work with dynamic

streams. On the other hand, only those functions specific to dynamic streams are in this suite.

**TABLE 78: AEGP\_DYNAMICSTREAMSUITE4**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewStreamRefForLayer | Retrieves the AEGP\_StreamRefH corresponding to the layer. This function is used to initiate a recursive walk of the layer’s streams.  AEGP\_GetNewStreamRefForLayer( AEGP\_PluginID aegp\_plugin\_id, AEGP\_LayerH layerH, AEGP\_StreamRefH \*streamPH); |
| AEGP\_GetNewStreamRefForMask | Retrieves the AEGP\_StreamRefH corresponding to the mask.  AEGP\_GetNewStreamRefForMask( AEGP\_PluginID aegp\_plugin\_id, AEGP\_MaskRefH maskH, AEGP\_StreamRefH \*streamPH); |
| AEGP\_GetStreamDepth | Retrieves the number of sub-streams associated with the given AEGP\_StreamRefH. The initial layer has a  depth of 0.  AEGP\_GetStreamDepth( AEGP\_StreamRefH streamH, A\_long \*depthPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetStreamGroupingType | Retrieves the grouping type for the given  AEGP\_StreamRefH.  AEGP\_GetStreamGroupingType( AEGP\_StreamRefH streamH, AEGP\_StreamGroupingType  \*group\_typeP);  AEGP\_StreamGroupingType will be one of the following:  AEGP\_StreamGroupingType\_NONE AEGP\_StreamGroupingType\_LEAF AEGP\_StreamGroupingType\_NAMED\_GROUP AEGP\_StreamGroupingType\_INDEXED\_GROUP |
| AEGP\_GetNumStreamsInGroup | Retrieves the number of streams associated with the  given AEGP\_StreamRefH. This function will return an error if called with an AEGP\_StreamRefH with  type AEGP\_StreamGroupingType\_LEAF.  AEGP\_GetNumStreamsInGroup( AEGP\_StreamRefH streamH,  A\_long \*num\_streamsPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetDynamicStreamFlags | Retrieves the flags for a given AEGP\_StreamRefH.  AEGP\_GetDynamicStreamFlags( AEGP\_StreamRefH streamH, AEGP\_DynStreamFlags \*flagsP);  AEGP\_DynStreamFlags will be one of the following:  AEGP\_DynStreamFlag\_ACTIVE\_EYEBALL AEGP\_DynStreamFlag\_HIDDEN AEGP\_DynStreamFlag\_DISABLED AEGP\_DynStreamFlag\_ELIDED AEGP\_DynStreamFlag\_SHOWN\_WHEN\_EMPTY AEGP\_DynStreamFlag\_SKIP\_REVEAL\_WHEN  \_UNHIDDEN  AEGP\_DynStreamFlag\_ACTIVE\_EYEBALL means that the stream is available for reading and writing.  AEGP\_DynStreamFlag\_HIDDEN means that, while the stream is still readable/writable, it may not currently be visible in the UI.  AEGP\_DynStreamFlag\_DISABLED A read-only flag. Indicates whether the AEGP\_StreamRefH is grayed out in the UI. Note that as of CS5, this flag will not be returned if a parameter is disabled. Instead, check PF\_PUI\_DISABLED in [ui\_flags](#_bookmark212).  AEGP\_DynStreamFlag\_ELIDED A read-only flag. Indicates that the AEGP\_StreamRefH is read-only, the user never sees it. However, the children are still seen and not indented in the Timeline panel.  AEGP\_DynStreamFlag\_SHOWN\_WHEN\_EMPTY New in CS6. A read-only flag. Indicates that this stream group should be shown when empty.  AEGP\_DynStreamFlag\_SKIP\_REVEAL\_WHEN  \_UNHIDDEN New in CS6. A read-only flag. Indicates that this stream property will not be automatically  revealed when un-hidden. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetDynamicStreamFlag | Sets the specified flag for the AEGP\_StreamRefH. Note; flags must be set individually. Undoable if undoableB is TRUE.  AEGP\_SetDynamicStreamFlag( AEGP\_StreamRefH streamH, AEGP\_DynStreamFlags one\_flag, A\_Boolean undoableB,  A\_Boolean setB);  This call may be used to dynamically show or hide parameters, by setting and clearing AEGP\_DynStreamFlag\_HIDDEN. However, AEGP\_DynStreamFlag\_DISABLED may not be set. |
| AEGP\_GetNewStreamRefByIndex | Retrieves a sub-stream by index from a given AEGP\_StreamRefH. Cannot be used on streams of type AEGP\_StreamGroupingType\_LEAF.  AEGP\_GetNewStreamRefByIndex( AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH parent\_groupH, A\_long indexL, AEGP\_StreamRefH \*streamPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewStreamRefByMatchname | Retrieves a sub-stream by match name from a given AEGP\_StreamRefH. Only legal for AEGP\_StreamGroupingType\_NAMED\_GROUP.  AEGP\_GetNewStreamRefByMatchname( AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH parent\_groupH, const A\_char \*match\_nameZ, AEGP\_StreamRefH \*streamPH);  Here are some handy stream names, for which references may be retrieved:  AEGP\_StreamGroupName\_MASK\_PARADE AEGP\_StreamGroupName\_MASK\_ATOM AEGP\_StreamName\_MASK\_FEATHER AEGP\_StreamName\_MASK\_OPACITY AEGP\_StreamName\_MASK\_OFFSET AEGP\_StreamGroupName\_EFFECT\_PARADE AEGP\_StreamGroupName\_LAYER AEGP\_StreamGroupName\_AV\_LAYER AEGP\_StreamGroupName\_TEXT\_LAYER AEGP\_StreamGroupName\_CAMERA\_LAYER AEGP\_StreamGroupName\_LIGHT\_LAYER AEGP\_StreamGroupName\_AUDIO AEGP\_StreamGroupName\_MATERIAL\_OPTIONS AEGP\_StreamGroupName\_TRANSFORM AEGP\_StreamGroupName\_LIGHT\_OPTIONS AEGP\_StreamGroupName\_CAMERA\_OPTIONS |
| AEGP\_DeleteStream | Deletes the specified stream from a stream grouping. Note that the caller must still dispose of any AEGP\_StreamRefH it’s already acquired (allocated) via the API. Undoable. Only valid for children of type [AEGP\_StreamGroupingType\_INDEXED\_GROUP](#_bookmark635).  AEGP\_DeleteStream( AEGP\_StreamRefH streamH);  Note: as of 6.5, if a stream is deleted while it or any child stream is selected, the current composition selection will become NULL. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_ReorderStream | Sets the new index of the specified AEGP\_StreamRefH. Undoable. Only valid for children of  AEGP\_StreamGroupingType\_INDEXED\_GROUP. The AEGP\_StreamRefH is updated to refer to the newly-ordered stream.  AEGP\_ReorderStream( AEGP\_StreamRefH streamH A\_long new\_indexL); |
| AEGP\_DuplicateStream | Duplicates the specified stream and appends it to the stream group. Undoable. Only valid for children of AEGP\_StreamGroupingType\_INDEXED\_GROUP.  AEGP\_DuplicateStream(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH streamH, A\_long \*new\_indexPL0); |
| AEGP\_SetStreamName | Sets the name of the given AEGP\_StreamRefH. Undoable. nameZ points to a null terminated UTF-16 string. Only valid for children of AEGP\_StreamGroupingType\_INDEXED\_GROUP. NOTE: If you retrieve the name with force\_englishB set to TRUE, you will get the  canonical, UNchanged name of the stream.  AEGP\_SetStreamName( AEGP\_StreamRefH streamH, const A\_UTF16Char \*nameZ);  Note: Use this on an effect stream’s group to change the display name of an effect. |
| AEGP\_CanAddStream | Returns whether or not it is currently possible to add a stream through the API.  AEGP\_CanAddStream(  AEGP\_StreamRefH group\_streamH, const A\_char \*match\_nameZ,  A\_Boolean \*can\_addPB); |
| AEGP\_AddStream | Adds a stream to the specified stream group. Undoable. Only valid for AEGP\_StreamGroupingType\_INDEXED\_GROUP.  AEGP\_AddStream(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_StreamRefH indxd\_grp\_streamH, const A\_char \*match\_nameZ, AEGP\_StreamRefH \*streamPH0); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetMatchName | Retrieves the match name for the specified AEGP\_StreamRefH. Note that this may differ from the display name, which can be retrieves using [AEGP\_GetStreamName](#_bookmark628), in [*AEGP\_StreamSuite*](#_bookmark622). nameZ can be up to AEGP\_MAX\_STREAM\_MATCH\_NAME\_SIZE in  length.  AEGP\_GetMatchName( AEGP\_StreamRefH streamH, A\_char \*nameZ); |
| AEGP\_GetNewParentStreamRef | Retrieves an AEGP\_StreamRefH for the parent of the specified AEGP\_StreamRefH.  AEGP\_GetNewParentStreamRef( AEGP\_PluginID plugin\_id, AEGP\_StreamRefH streamH, AEGP\_StreamRefH \*parentPH); |
| AEGP\_GetStreamIsModified | Returns whether or not the specified AEGP\_StreamRefH has been modified. Note: the same result is available throught the After Effect user  interface by typing “UU” with the composition selected.  AEGP\_GetStreamIsModified( AEGP\_StreamRefH streamH, A\_Boolean \*modifiedPB); |
| AEGP\_GetStreamIndexInParent | Retrieves the index of a given stream, relative to its parent stream. Only valid for children of AEGP\_StreamGroupingType\_INDEXED\_GROUP  AEGP\_GetStreamIndexInParent( AEGP\_StreamRefH streamH, A\_long \*indexPL);  NOTE: As mentioned *elsewhere*, AEGP\_StreamRefHs don’t persist across function calls. If streams are re-ordered, added or removed, all AEGP\_StreamRefHs previously retrieved may be invalidated. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_IsSeparationLeader | Valid on leaf streams only. Returns true if this stream is a multidimensional stream that can have its dimensions separated, though they may not be currently separated.  Terminology: A Leader is the stream that can be separated, a Follower is one of N automatic streams that correspond to the N dimensions of the Leader.  A Leader isn't always separated, call AEGP\_AreDimensionsSeparated to find out if it is. As of CS4, the only stream that is ever separarated is the layer's Position property. Please \*do not\* write code assuming that, we anticipate allowing separation of more streams in the future.  AEGP\_IsSeparationLeader( AEGP\_StreamRefH streamH, A\_Boolean \*leaderPB); |
| AEGP\_AreDimensionsSeparated | [Methods such as](#_bookmark643) AEGP\_GetNewKeyframeValue that work on keyframe indices will most definitely \*not\* work on the Leader property, you will need to retrieve and operate on the Followers explicitly.  AEGP\_AreDimensionsSeparated( AEGP\_StreamRefH streamH, A\_Boolean \*separatedPB); |
| AEGP\_SetDimensionsSeparated | Valid only if AEGP\_IsSeparationLeader() is true.  AEGP\_AreDimensionsSeparated( AEGP\_StreamRefH streamH, A\_Boolean \*separatedPB); |
| AEGP\_GetSeparationFollower | Retrieve the Follower stream corresponding to a given dimension of the Leader stream. dimS can range from 0 to AEGP\_GetStreamValueDimensionality(lea der\_streamH) - 1.  AEGP\_GetSeparationFollower( AEGP\_StreamRefH leader\_streamH A\_short dimS, AEGP\_StreamRefH \*follower\_streamPH); |
| AEGP\_IsSeparationFollower | Valid on leaf streams only. Returns true if this stream is a one dimensional property that represents one of the dimensions of a Leader. You can retrieve stream from the Leader using AEGP\_GetSeparationFollower().  AEGP\_IsSeparationFollower( AEGP\_StreamRefH streamH A\_Boolean \*followerPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetSeparationLeader | Valid on separation Followers only, returns the Leader it is part of.  AEGP\_GetSeparationLeader( AEGP\_StreamRefH follower\_streamH, AEGP\_StreamRefH \*leader\_streamPH); |
| AEGP\_GetSeparationDimension | Valid on separation Followers only, returns which dimension of the Leader it corresponds to.  AEGP\_GetSeparationDimension( AEGP\_StreamRefH follower\_streamH, A\_short \*dimPS); |

**WORKING WITH KEYFRAMES**

Keyframes make After Effects what it is. AEGPs (and...ssshh, don’t tell anyone...effects) can use this suite to add, manipulate and remove keyframes from any keyframe-able stream.

**TABLE 79: AEGP\_KEYFRAMESUITE3**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetStreamNumKFs | Retrieves the number of keyframes on the given stream. Returns AEGP\_NumKF\_NO\_DATA if the stream is not keyframe-able. Also, note that a stream without  keyframes isn’t necessarily constant; it can be altered by expressions.  AEGP\_GetStreamNumKFs( AEGP\_StreamRefH streamH, A\_long \*num\_kfsPL); |
| AEGP\_GetKeyframeTime | Retrieves the time of the specified keyframe.  AEGP\_GetKeyframeTime( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex index, AEGP\_LTimeMode time\_mode,  A\_Time \*timePT); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_InsertKeyframe | Adds a keyframe to the specified stream (at the specified composition or layer time). Returns the new keyframe’s  index. All indexes greater than the new index are now  invalid (but you knew that). If there is already a keyframe at that time, the values will be updated.  AEGP\_InsertKeyframe( AEGP\_StreamRefH streamH,  AEGP\_LTimeMode time\_mode,  const A\_Time \*timePT, AEGP\_KeyframeIndex \*key\_indexP); |
| AEGP\_DeleteKeyframe | Deletes the specified keyframe.  AEGP\_DeleteKeyframe( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index); |
| AEGP\_GetNewKeyframeValue | Creates and populates an AEGP\_StreamValue2 for the stream’s value at the time of the keyframe. The  returned AEGP\_StreamValue2 must be disposed of using AEGP\_DisposeStreamValue.  AEGP\_GetNewKeyframeValue( AEGP\_PluginID plugin\_id, AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index, AEGP\_StreamValue2 \*valueP); |
| AEGP\_SetKeyframeValue | Sets the stream’s value at the time of the keyframe.  AEGP\_SetKeyframeValue( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex index, const AEGP\_StreamValue2\*valP); |
| AEGP\_GetStreamValue Dimensionality | Retrieves the dimensionality of the stream’s value.  AEGP\_GetStreamValueDimensionality( AEGP\_StreamRefH streamH, A\_short \*value\_dimPS); |
| AEGP\_GetStreamTemporal Dimensionality | Retrieves the temporal dimensionality of the stream.  AEGP\_GetStreamTemporalDimensionality( AEGP\_StreamRefH streamH,  A\_short \*t\_dimPS); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewKeyframe SpatialTangents | Returns the AEGP\_StreamValue2s representing the stream’s tangential values at the time of the keyframe.  The returned AEGP\_StreamValue2s must be  disposed of using AEGP\_DisposeStreamValue.  AEGP\_GetNewKeyframeSpatialTangents( AEGP\_PluginID plugin\_id, AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index, AEGP\_StreamValue2 \*in\_tanP0, AEGP\_StreamValue2 \*out\_tanP0); |
| AEGP\_SetKeyframeSpatialTangents | Specifies the tangential AEGP\_StreamValue2s to be used for the stream’s value at the time of the  keyframe. The AEGP\_StreamValue2s passed for in and out tangents are not adopted by After Effects, and must be disposed of using AEGP\_DisposeStreamValue.  AEGP\_SetKeyframeSpatialTangents( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index, const AEGP\_StreamValue2\*in\_tP0, const AEGP\_StreamValue2\*out\_tP0);  NOTE: In AEGP\_KeyframeSuite2 and prior versions, the values returned from this function were wrong when called on an effect point control stream or anchor point.  They were not multiplied by the layer size. Now they are. |
| AEGP\_GetKeyframeTemporalEase | Retrieves the AEGP\_KeyframeEases associated with the specified dimension of the stream’s value at the time of the keyframe. dimensionL ranges from 0 to (temporal\_dimensionality -1).  AEGP\_GetKeyframeTemporalEase( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndexkey\_index, A\_long dimensionL, AEGP\_KeyframeEase \*in\_easeP0, AEGP\_KeyframeEase \*out\_easeP0);  NOTE: the returned ease values must be multiplied by layer height to match the values displayed in the After Effects UI. |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetKeyframeTemporalEase | Specifies the AEGP\_KeyframeEases to be used for the stream’s value at the time of the keyframe. dimensionL ranges from 0 to (temporal\_dimensionality -1). The AEGP\_KeyframeEases passed are not adopted by After Effects.  AEGP\_SetKeyframeTemporalEase( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index, A\_long dimL,  const AEGP\_KeyframeEase \*in\_P0, const AEGP\_KeyframeEase \*outP0); |
| AEGP\_GetKeyframeFlags | Retrieves the flags currently set for the keyframe.  AEGP\_GetKeyframeFlags( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index, AEGP\_KeyframeFlags \*flagsP);  \*flagsP will be a combination of the following:  AEGP\_KeyframeFlag\_NONE AEGP\_KeyframeFlag\_TEMPORAL\_CONTINUOUS AEGP\_KeyframeFlag\_TEMPORAL\_AUTOBEZIER AEGP\_KeyframeFlag\_SPATIAL\_CONTINUOUS AEGP\_KeyframeFlag\_SPATIAL\_AUTOBEZIER AEGP\_KeyframeFlag\_ROVING |
| AEGP\_SetKeyframeFlag | Sets the specified flag for the keyframe. Flags must be set individually.  AEGP\_SetKeyframeFlag( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index, AEGP\_KeyframeFlags flag, A\_Boolean valueB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetKeyframeInterpolation | Retrieves the in and out AEGP\_KeyframeInterpolationTypes for the specified keyframe.  AEGP\_GetKeyframeInterpolation( AEGP\_StreamRefH streamH, AEGP\_KeyframeIndex key\_index, AEGP\_KeyframeInterpolationType  \*inP0, AEGP\_KeyframeInterpolationType  \*outP0);  AEGP\_KeyframeInterpolationType is one of the following:  AEGP\_KeyInterp\_NONE AEGP\_KeyInterp\_LINEAR AEGP\_KeyInterp\_BEZIER AEGP\_KeyInterp\_HOLD |
| AEGP\_SetKeyframeInterpolation | Specifies the in and out AEGP\_KeyframeInterpolationTypes to be used for the given keyframe.  AEGP\_SetKeyframeInterpolation( AEGP\_StreamRefHstreamH, AEGP\_KeyframeIndex key\_index,  AEGP\_KeyframeInterpolationType  in\_interp,  AEGP\_KeyframeInterpolationType  out\_interp); |
| AEGP\_StartAddKeyframes | Informs After Effects that you’re going to be adding several keyframes to the specified stream. After Effects will return an allocated opaque AEGP\_AddKeyframesInfoH, for use with the calls below.  AEGP\_StartAddKeyframes( AEGP\_StreamRefH streamH, AEGP\_AddKeyframesInfoH\*akPH); |
| AEGP\_AddKeyframes | Adds a keyframe to the specified stream at the specified (layer or composition) time. Note: this doesn’t actually do anything to the stream’s value.  AEGP\_AddKeyframes( AEGP\_AddKeyframesInfoH akH, AEGP\_LTimeMode time\_mode,  const A\_Time \*timePT,  A\_long \*indexPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetAddKeyframe | Sets the value of the specified keyframe.  AEGP\_SetAddKeyframe( AEGP\_AddKeyframesInfoH akH, A\_long indexL,  const AEGP\_StreamValue2\*valueP); |
| AEGP\_EndAddKeyframes | Tells After Effects you’re done adding keyframes.  AEGP\_EndAddKeyframes( A\_Boolean addB, AEGP\_AddKeyframesInfoH akH); |

**ADDING MULTIPLE KEYFRAMES**

Each time you call [AEGP\_InsertKeyframe](#_bookmark642)(), the entire stream is added to the undo

stack. If you’re adding one or two keyframes, this isn’t a problem. However, if you’re writing a keyframer, you’ll want to do things the *right* way.

Before you begin adding keyframes, call the (very-appropriately-named) [AEGP\_StartAddKeyframes](#_bookmark644), passing it an opaque AEGP\_AddKeyframesInfoH. For each keyframe to add, call [AEGP\_AddKeyframes](#_bookmark645) to set the time to be used (and get the newly- added keyframe’s index), then [AEGP\_SetAddKeyframe](#_bookmark646) to specify the value to be used.

Once you’re finished, call [AEGP\_EndAddKeyframes](#_bookmark647) to let know After Effects know it’s time to add the changed parameter stream to the undo stack.

**MARKER STREAMS**

AEGP\_MarkerSuite allows for direct manipulation of marker data.

**TABLE 80: AEGP\_MARKERSUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewMarker | Creates a new marker.  AEGP\_NewMarker(  AEGP\_MarkerValP \*markerPP); |
| AEGP\_DisposeMarker | Disposes of a marker.  AEGP\_DisposeMarker( AEGP\_MarkerValP markerP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_DuplicateMarker | Duplicates a marker (didn’t see *that* one coming, eh?).  AEGP\_DuplicateMarker( AEGP\_MarkerValP markerP, AEGP\_MarkerValP \*new\_markerP); |
| AEGP\_SetMarkerFlag | Sets a marker flag’s value.  AEGP\_SetMarkerFlag( AEGP\_MarkerValP markerP, |
| AEGP\_MarkerFlagType flagType,  A\_Boolean valueB);  Currently, AEGP\_MarkerFlagType is one of the following:  AEGP\_MarkerFlag\_NONE AEGP\_MarkerFlag\_NAVIGATION |
| AEGP\_GetMarkerFlag | Gets the value (see above) of a given  AEGP\_MarkerFlagType. AEGP\_GetMarkerFlag(  AEGP\_ConstMarkerValP markerP, AEGP\_MarkerFlagType flagType A\_Boolean \*valueBP); |
| AEGP\_GetMarkerString | Retrieves the UTF-16, NULL-terminated string located in the specified marker field. Must be disposed of by caller using AEGP\_FreeMemHandle.  AEGP\_GetMarkerString( AEGP\_PluginID id, AEGP\_ConstMarkerValP markerP, AEGP\_MarkerStringType strType,  AEGP\_MemHandle \*unicodePH); |
| AEGP\_SetMarkerString | Sets the specified field of a marker to the provided text.  AEGP\_SetMarkerString( AEGP\_MarkerValP markerP, AEGP\_MarkerStringType strType,  const A\_u\_short \*unicodeP,  A\_long lengthL); |
| AEGP\_CountCuePointParams | Returns the number of cue point parameters.  AEGP\_CountCuePointParams( AEGP\_ConstMarkerValP markerP, A\_long \*paramsLP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetIndCuePointParam | Returns the cue point param at the specified index (which must be between 0 and (cue point params -1). Returned  handles are UTF-16, NULL-terminated strings, and must be disposed of by caller using AEGP\_FreeMemHandle.  AEGP\_GetIndCuePointParam( AEGP\_PluginID id, AEGP\_ConstMarkerValP markerP,  A\_long param\_indexL,  AEGP\_MemHandle \*unicodeKeyPH,  AEGP\_MemHandle \*uni\_ValuePH); |
| AEGP\_SetIndCuePointParam | Set the value of an indexed cue point parameter to the specified value. key\_lengthL is “number of unicode  characters”, and value\_lenL is the length of the provided value. unicode\_KeyP and unicode\_ValueP point to UTF-16 data.  AEGP\_SetIndCuePointParam( AEGP\_MarkerValP markerP, A\_long param\_idxL, const A\_u\_short \*unicode\_KeyP, A\_long key\_lengthL,  const A\_u\_short \*unicode\_ValueP, A\_long value\_lengthL); |
| AEGP\_InsertCuePointParam | Inserts a cue point parameter. This call is following by AEG- P\_SetIndCuePointParam to actually set the data.  AEGP\_InsertCuePointParam( AEGP\_MarkerValP markerP, A\_long param\_idxL); |
| AEGP\_DeleteInd CuePointParam | Deletes the cue point param at the specified index.  AEGP\_DeleteIndCuePointParam( AEGP\_MarkerValP markerP, A\_long param\_idxL); |
| AEGP\_SetMarkerDuration | AEGP\_SetMarkerDuration( AEGP\_MarkerValP markerP, const A\_Time \*durationPT); |
| AEGP\_GetMarkerDuration | AEGP\_GetMarkerDuration( AEGP\_ConstMarkerValP markerP, A\_Time \*durationPT); |

Access, manipulate, and delete a layer’s masks.

**TABLE 81: AEGP\_MASKSUITE6**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLayerNumMasks | Counts the masks applied to a layer,  AEGP\_GetLayerNumMasks(  AEGP\_LayerH aegp\_layerH,  A\_long \*num\_masksPL); |
| AEGP\_GetLayerMaskBy Index | Given a layer handle and mask index, returns a pointer to the mask handle. You must destroy the mask handle by using [AEGP\_DisposeMask()](#_bookmark656).  AEGP\_GetLayerMaskByIndex( AEGP\_LayerH aegp\_layerH,  A\_long mask\_indexL,  AEGP\_MaskRefH \*maskPH); |
| AEGP\_DisposeMask | Dispose of a mask handle.  AEGP\_DisposeMask(  AEGP\_MaskRefH maskH); |
| AEGP\_GetMaskInvert | Given a mask handle, determines if the mask is inverted or not.  AEGP\_GetMaskInvert( AEGP\_MaskRefH maskH,  A\_Boolean \*invertPB); |
| AEGP\_SetMaskInvert | Sets the inversion state of a mask.  AEGP\_SetMaskInvert)(  AEGP\_MaskRefH mask\_refH,  A\_Boolean invertB); |
| AEGP\_GetMaskMode | Given a mask handle, returns the current mode of the mask. PF\_MaskMode\_NONE does nothing, PF\_MaskMode\_ADD is the default behavior.  PF\_MaskMode\_NONE PF\_MaskMode\_ADD, PF\_MaskMode\_SUBTRACT, PF\_MaskMode\_INTERSECT, PF\_MaskMode\_LIGHTEN, PF\_MaskMode\_DARKEN, PF\_MaskMode\_DIFFERENCE,  AEGP\_GetMaskMode( AEGP\_MaskRefH maskH,  PF\_MaskMode \*modeP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetMaskMode | Sets the mode of the given mask.  AEGP\_SetMaskMode( AEGP\_MaskRefH maskH,  PF\_MaskMode mode); |
| AEGP\_GetMaskMotionBlurState | Retrieves the motion blur setting for the given mask.  AEGP\_GetMaskMotionBlurState( AEGP\_MaskRefH mask\_refH,  AEGP\_MaskMBlur \*blur\_stateP); AEGP\_MaskMBlur will be one of the following:  AEGP\_MaskMBlur\_SAME\_AS\_LAYER AEGP\_MaskMBlur\_OFF AEGP\_MaskMBlur\_ON |
| AEGP\_SetMaskMotionBlurState | New in CS6. Sets the motion blur setting for the given mask.  AEGP\_SetMaskMotionBlurState( AEGP\_MaskRefH mask\_refH, AEGP\_MaskMBlur blur\_state); |
| AEGP\_GetMaskFeatherFalloff | New in CS6. Gets the type of feather falloff for the given mask, either AEGP\_MaskFeatherFalloff\_SMOOTH or AEGP\_MaskFeatherFalloff\_LINEAR.  AEGP\_SetMaskMotionBlurState( AEGP\_MaskRefH mask\_refH, AEGP\_MaskFeatherFalloff  \*feather\_falloffP); |
| AEGP\_SetMaskFeatherFalloff | Sets the type of feather falloff for the given mask.  AEGP\_SetMaskMotionBlurState( AEGP\_MaskRefH mask\_refH,  AEGP\_MaskFeatherFalloff  feather\_falloff); |
| AEGP\_GetMaskName | Removed in CS4. Use [AEGP\_GetNewStreamRefForMask](#_bookmark634)  and the name functions in the Dynamic Stream Suite instead. |
| AEGP\_SetMaskName |
| AEGP\_GetMaskID | Retrieves the AEGP\_MaskIDVal for the given  AEGP\_MaskRefH, for use in uniquely identifying the mask.  AEGP\_GetMaskID(  AEGP\_MaskRefH mask\_refH,  AEGP\_MaskIDVal \*id\_valP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_CreateNewMask | Creates a new mask on the referenced AEGP\_LayerH, with zero nodes. The new mask’s index is returned.  AEGP\_CreateNewMask(  AEGP\_LayerH layerH,  AEGP\_MaskRefH \*mask\_refPH,  A\_long \*mask\_indexPL0); |
| AEGP\_DeleteMaskFromLayer | AEGP\_DeleteMaskFromLayer( AEGP\_MaskRefH mask\_refH);  NOTE: As of 6.5, if you delete a mask and it or a child stream is selected, the current selection within the composition will become NULL. |
| AEGP\_GetMaskColor | Retrieves the color of the specified mask.  AEGP\_GetMaskColor(  AEGP\_MaskRefH mask\_refH,  AEGP\_ColorVal \*colorP); |
| AEGP\_SetMaskColor | Sets the color of the specified mask.  AEGP\_SetMaskColor(  AEGP\_MaskRefH mask\_refH, const AEGP\_ColorVal \*colorP); |
| AEGP\_GetMaskLockState | Retrieves the lock state of the specified mask.  AEGP\_GetMaskLockState( AEGP\_MaskRefH mask\_refH,  A\_Boolean \*is\_lockedPB); |
| AEGP\_SetMaskLockState | Sets the lock state of the specified mask.  AEGP\_SetMaskLockState( AEGP\_MaskRefH mask\_refH,  A\_Boolean lockB); |
| AEGP\_GetMaskIsRotoBezier | Returns whether or not the given mask is used as a rotobezier.  AEGP\_GetMaskIsRotoBezier( AEGP\_MaskRefH mask\_refH,  A\_Boolean \*is\_roto\_bezierPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetMaskIsRotoBezier | Sets whether a given mask is to be used as a rotobezier.  AEGP\_SetMaskIsRotoBezier( AEGP\_MaskRefH mask\_refH,  A\_Boolean \*is\_roto\_bezierPB); |
| AEGP\_DuplicateMask | Duplicates a given AEGP\_MaskRefH. Caller must dispose of duplicate.  AEGP\_DuplicateMask(  AEGP\_MaskRefH orig\_mask\_refH,  AEGP\_MaskRefH \*dupe\_mask\_refPH); |

**MASK OUTLINES**

The Mask Suite above tells plug-ins about the masks on a layer, but not about the details of those masks. This is because processing is required on After Effects’ part to access the

information; the information isn’t just lying around. Plug-ins access that information using this Mask Outline Suite.

**TABLE 82: AEGP\_MASKOUTLINESUITE3**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_IsMaskOutlineOpen | Given an mask outline pointer (obtainable through [AEGP\_StreamSuite](#_bookmark622)), determines if the mask path is open or closed.  AEGP\_IsMaskOutlineOpen( AEGP\_MaskOutlineVal \*mask\_outlineP, A\_Boolean \*openPB); |
| AEGP\_SetMaskOutlineOpen | Sets the open state of the given mask outline.  AEGP\_SetMaskOutlineOpen( AEGP\_MaskOutlineValH mask\_outlineH, A\_Boolean openB); |
| AEGP\_GetMaskOutlineNum Segments | Given a mask outline pointer, returns the number of segments in the path. num\_segmentsPL is the total number of segments [0...N-1].  AEGP\_GetMaskOutlineNumSegments( AEGP\_MaskOutlineVal \*mask\_outlineP, A\_long \*num\_segmentsPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetMaskOutline VertexInfo | Given a mask outline pointer and a point between 0 and the total number of segments. For closed mask paths, vertex[0] is the same as vertex[num\_segments].  AEGP\_GetMaskOutlineVertexInfo( AEGP\_MaskOutlineVal \*mask\_outlineP, A\_long which\_pointL, AEGP\_MaskVertex \*vertexP); |
| AEGP\_SetMaskOutlineVertex Info | Sets the vertex information for a given index. Setting vertex 0 is special; its in tangent will actually set the out tangent of the last vertex in the outline. Of course, which\_pointL must be valid for the mask outline, or the function will return an error.  AEGP\_SetMaskOutlineVertexInfo( AEGP\_MaskOutlineValH mask\_outlineH, AEGP\_VertexIndex which\_pointL, AEGP\_MaskVertex \*vertexP); |
| AEGP\_CreateVertex | Creates a vertex at index position. All vertices which formerly had an AEGP\_VertexIndex of position or greater will have their  indices incremented by one.  AEGP\_CreateVertex(  AEGP\_MaskOutlineValH mask\_outlineH, AEGP\_VertexIndex position);.  NOTE: All masks must have at least one vertex. |
| AEGP\_DeleteVertex | Removes a vertex from a mask.  AEGP\_DeleteVertex(  AEGP\_MaskOutlineValH mask\_outlineH, AEGP\_VertexIndex index); |
| AEGP\_GetMaskOutlineNum Feathers | New in CS6.  AEGP\_DeleteVertex(  AEGP\_MaskOutlineValH mask\_outlineH, A\_long \*num\_feathersPL); |
| AEGP\_GetMaskOutline FeatherInfo | New in CS6.  AEGP\_GetMaskOutlineFeatherInfo( AEGP\_MaskOutlineValH mask\_outlineH, AEGP\_FeatherIndex which\_featherL, AEGP\_MaskFeather \*featherP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetMaskOutline FeatherInfo | New in CS6. Feather must already exist; use  AEGP\_CreateMaskOutlineFeather first, if needed.  AEGP\_SetMaskOutlineFeatherInfo( AEGP\_MaskOutlineValH mask\_outlineH, AEGP\_VertexIndex which\_featherL, const AEGP\_MaskFeather \*featherP); |
| AEGP\_CreateMaskOutline Feather | New in CS6. Index of new feather is passed back in  insert\_positionP.  AEGP\_CreateMaskOutlineFeather( AEGP\_MaskOutlineValH mask\_outlineH, const AEGP\_MaskFeather \*featherP0, AEGP\_FeatherIndex \*insert\_positionP); |
| AEGP\_DeleteMaskOutline Feather | New in CS6.  AEGP\_DeleteMaskOutlineFeather( AEGP\_MaskOutlineValH mask\_outlineH, AEGP\_FeatherIndex index); |

**MASK FEATHERING**

New for CS6, masks can be feathered. AEGP\_MaskFeather is defined as follows:

typedef struct {

A\_long segment; // mask segment where feather is

PF\_FpLong segment\_sF; // 0-1: feather location on segment

PF\_FpLong radiusF; // negative value allowed if type == AEGP\_MaskFeatherType\_INNER

PF\_FpShort ui\_corner\_angleF; // 0-1: angle of UI handle on corners

PF\_FpShort tensionF; // 0-1: tension of boundary at feather pt

AEGP\_MaskFeatherInterp interp; AEGP\_MaskFeatherType type;

} AEGP\_MaskFeather;

AEGP\_MaskFeatherInterp is either AEGP\_MaskFeatherInterp\_NORMAL or

AEGP\_MaskFeatherInterp\_HOLD\_CW.

AEGP\_MaskFeatherType is either AEGP\_MaskFeatherType\_OUTER or

AEGP\_MaskFeatherType\_INNER.

**WORKING WITH TEXT LAYERS**

This suite enables AEGPs to get and set the text associated with text layers. Note: to get started, retrieve an AEGP\_TextDocumentH by calling [AEGP\_GetLayerStreamValue](#_bookmark631), above, and passing AEGP\_StreamType\_TEXT\_DOCUMENT as the AEGP\_StreamType.

**TABLE 83: AEGP\_TEXTDOCUMENTSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewText | Retrieves the UTF-16, NULL-terminated string used in the AEGP\_TextDocumentH. Note: After Effects will allocate the AEGP\_MemHandle; your plug-in must  dispose of it when done using  AEGP\_FreeMemHandle.  AEGP\_GetNewText( AEGP\_PluginID id,  AEGP\_TextDocumentH text\_docH, AEGP\_MemHandle \*unicodePH); |
| AEGP\_SetText | Specifies the text to be used by the  AEGP\_TextDocumentH.  AEGP\_SetText(  AEGP\_TextDocumentH text\_docH, const A\_u\_short \*unicodePS,  long lengthL); |

**WORKING WITH TEXT OUTLINES**

The AEGP\_TextLayerSuite provides access to the actual outlines of the text used by text layers. One you have a path, you can manipulate it with [PF\_PathQuerySuite](#_bookmark372) and [PF\_PathDataSuite](#_bookmark369) .

**TABLE 84: AEGP\_TEXTLAYERSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewTextOutlines | Allocates and returns a handle to the  AEGP\_TextOutlinesHs associated with the  specified layer. outlinesPH will be NULL if there are no AEGP\_TextOutlinesHs associated with layerH (in other words, if it’s not a text layer).  AEGP\_GetNewTextOutlines(  AEGP\_LayerH layerH, /  const A\_Time \*layer\_timePT, AEGP\_TextOutlinesH \*outlinesPH); |
| AEGP\_DisposeTextOutlines | Dispose of those outlines we allocated on your behalf!  AEGP\_DisposeTextOutlines( AEGP\_TextOutlinesH outlinesH); |
| AEGP\_GetNumTextOutlines | Retrieves the number of text outlines for the layer.  AEGP\_GetNumTextOutlines( AEGP\_TextOutlinesH outlinesH, A\_long \*num\_otlnsPL); |
| AEGP\_GetIndexedTextOutline | Returns a PF\_PathOutlinePtr for the specifed text outline.  AEGP\_GetIndexedTextOutline( AEGP\_TextOutlinesH outlinesH, A\_long path\_indexL, PF\_PathOutlinePtr \*pathPP); |

The Utility suite supplies error message handling, AEGP version checking and access to the undo stack. Everything you need to keep After Effects and your plug-in tidy.

**TABLE 85: AEGP\_UTILITYSUITE6**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_ReportInfo | Displays dialog with name of the AEGP followed by the string passed.  AEGP\_ReportInfo(  AEGP\_PluginID aegp\_plugin\_id, const A\_char \*info\_stringZ); |
| AEGP\_ReportInfoUnicode | New in CC. Displays dialog with name of the AEGP followed by the unicode string passed.  AEGP\_ReportInfoUnicode(  AEGP\_PluginID aegp\_plugin\_id, const A\_UTF16Char \*info\_stringP); |
| AEGP\_GetDriverSpecVersion | Returns version of AEGPDriver plug-in (use to determine supported features).  AEGP\_GetDriverSpecVersion(  A\_short \*major\_versionPS,  A\_short \*minor\_versionPS); |
| AEGP\_StartQuietErrors | Silences errors. Must be balanced with AEGP\_EndQuietErrors. The AEGP\_ErrReportState is an opaque structure private to After Effects.  AEGP\_StartQuietErrors( AEGP\_ErrReportState \*err\_stateP); |
| AEGP\_EndQuietErrors | Re-enables errors.  AEGP\_EndQuietErrors( AEGP\_ErrReportState \*err\_stateP) |
| AEGP\_StartUndoGroup | Add action(s) to the undo queue. The user may undo any actions between this and AEGP\_EndUndoGroup(). The  undo\_nameZ will appear in the edit menu.  AEGP\_StartUndoGroup(  const A\_char \*undo\_nameZ); |
| AEGP\_EndUndoGroup | Ends the undo list.  AEGP\_EndUndoGroup(); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RegisterWithAEGP | Returns an AEGP\_PluginID, which effect plug-ins can then use in calls to many functions throughout the AEGP API. Effects should only call this function once, during PF\_Cmd\_GLOBAL\_SETUP, and save the AEGP\_PluginID for later use. The first parameter can be any value, and the second parameter should be the plug-in’s match name.  AEGP\_RegisterWithAEGP( AEGP\_GlobalRefcon global\_refcon, const A\_char \*plugin\_nameZ,  AEGP\_PluginID \*plugin\_id); |
| AEGP\_GetMainHWND | Retrieves After Effects’ HWND; useful when displaying your own dialog on Windows. If you don’t use After Effects’ HWND, your modal dialog will not prevent interaction with the windows  behind, and pain will ensue.  AEGP\_GetMainHWND(  void \*main\_hwnd); |
| AEGP\_ShowHideAllFloaters | Toggles whether or not floating palettes are displayed. Use this with care; users get twitchy when you unexpectedly change the UI on them.  AEGP\_ShowHideAllFloaters(  A\_Boolean include\_tool\_palB); |
| AEGP\_PaintPalGetForeColor | Retrieves the foreground color from the paint palette.  AEGP\_PaintPalGetForeColor( AEGP\_ColorVal \*fore\_colorP); |
| AEGP\_PaintPalGetBackColor | Retrieves the background color from the paint palette.  AEGP\_PaintPalGetBackColor( AEGP\_ColorVal \*back\_colorP); |
| AEGP\_PaintPalSetForeColor | Sets the foreground color in the paint palette.  AEGP\_PaintPalSetForeColor(  const AEGP\_ColorVal \*fore\_colorP); |
| AEGP\_PaintPalSetBackColor | Sets the background color in the paint palette.  AEGP\_PaintPalSetBackColor(  const AEGP\_ColorVal \*back\_colorP); |
| AEGP\_CharPalGetFillColor | Retrieves the fill color from the character palette.  AEGP\_CharPalGetFillColor(  A\_Boolean \*is\_fcolor\_definedPB,  AEGP\_ColorVal \*fill\_colorP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_CharPalGetStrokeColor | Retrieves the stroke color from the character palette.  AEGP\_CharPalGetStrokeColor(  A\_Boolean \*is\_scolor\_definedPB,  AEGP\_ColorVal \*stroke\_colorP); |
| AEGP\_CharPalSetFillColor | Sets the fill color in the character palette.  AEGP\_CharPalSetFillColor(  const AEGP\_ColorVal \*fill\_colorP); |
| AEGP\_CharPalSetStrokeColor | Sets the stroke color in the character palette.  AEGP\_CharPalSetStrokeColor(  const AEGP\_ColorVal \*stroke\_colorP); |
| AEGP\_CharPalIsFillColorUI Frontmost | Returns whether or not the fill color is frontmost. If it isn’t, the stroke color is frontmost.  AEGP\_CharPalIsFillColorUIFrontmost( A\_Boolean \*is\_fcolor\_selectedPB); |
| AEGP\_ConvertFpLongTo HSFRatio | Returns an A\_Ratio interpretation of the given A\_FpLong. Useful for horizontal scale factor interpretation.  AEGP\_ConvertFpLongToHSFRatio( A\_FpLong numberF,  A\_Ratio \*ratioPR); |
| AEGP\_ConvertHSFRatioTo FpLong | Returns an A\_FpLong interpretation of the given A\_Ratio. Useful for horizontal scale factor interpretation.  AEGP\_ConvertHSFRatioToFpLong( A\_Ratio ratioR,  A\_FpLong \*numberPF); |
| AEGP\_CauseIdleRoutines ToBeCalled | This routine is safe to call from threads other than the main  thread. It is asynchronous and will return before the idle handler is called. The suite functions to get this function pointer are not thread safe; save it off in the main thread for use by the child  thread.  AEGP\_CauseIdleRoutinesToBeCalled(void); |
| AEGP\_GetSuppress InteractiveUI | Returns whether After Effects is running without a user interface.  AEGP\_GetSuppressInteractiveUI(  A\_Boolean \*ui\_is\_suppressedPB); |
| AEGP\_WriteToOSConsole | Sends a string to the OS console.  AEGP\_WriteToOSConsole(  const A\_char \*textZ); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_WriteToDebugLog | Writes a message to the debug log, or to the OS command line if After Effects was launched with the “-debug” option.  AEGP\_WriteToDebugLog(  const A\_char \*subsystemZ,  const A\_char \*event\_typeZ,  const A\_char \*infoZ); |
| AEGP\_GetLastErrorMessage | Retrieves the last error message displayed to the user, and its  associated error number. Pass in the size of the character buffer to be returned.  AEGP\_GetLastErrorMessage(  A\_long buffer\_size,  A\_char \*error\_string,  A\_Err \*error\_num); |
| AEGP\_IsScriptingAvailable | Returns TRUE if scripting is available to the plug-in.  AEGP\_IsScriptingAvailable(  A\_Boolean \*outAvailablePB); |
| AEGP\_ExecuteScript | Have After Effects execute a script. The script passed in can be in either UTF-8 or the current application encoding (if platform\_encodingB is passed in as TRUE).  The two out arguments are optional. The value of the last line of the script is what is passed back in outResultPH0.  AEGP\_ExecuteScript(  AEGP\_PluginID inPlugin\_id,  const A\_char \*inScriptZ,  const A\_Boolean platform\_encodingB, AEGP\_MemHandle \*outResultPH0, AEGP\_MemHandle \*outErrStringPH0); |
| AEGP\_HostIsActivated | Returns TRUE if the user has successfully activated After Effects.  AEGP\_HostIsActivated(  A\_Boolean \*is\_activatedPB); |
| AEGP\_GetPluginPlatformRef | On Mac OS, returns a CFBundleRef to your Mach-O plug-in, or NULL for a CFM plug-in. Always returns NULL on Windows (you can use an OS-specific entry point to capture your DLLInstance).  AEGP\_GetPluginPlatformRef( AEGP\_PluginID plug\_id,  void \*\*plat\_refPPV); |
| AEGP\_UpdateFontList | Rescans the system font list.  AEGP\_UpdateFontList(); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetPluginPaths | New in CC. Returns a particular path associated with the plug-in:  AEGP\_GetPathTypes\_PLUGIN - (Not Implemented) The path to the location of the plug-in itself.  AEGP\_GetPathTypes\_USER\_PLUGIN -The suite specific location of user specific plug-ins.  AEGP\_GetPathTypes\_ALLUSER\_PLUGIN - The suite specific location of plug-ins shared by all users.  AEGP\_GetPathTypes\_APP - The After Effects .exe or .app location. Not plug-in specific.  AEGP\_GetPluginPaths(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_GetPathTypes path\_type AEGP\_MemHandle \*unicode\_pathPH); |

**PERSISTENT DATA SUITE**

Plug-ins have read and write access to persistent data in After Effects’ preferences. AEGPs may add and manage their own persistent data using the following suite. The data entries are accessed by (section key, value key) pairs. It is recommended that plug-ins use their

matchname as their section key, or as a prefix if using multiple section keys.

The available data types are A\_long, A\_FpLong, strings, and void\*. A\_FpLongs are stored with 6 decimal places of precision. There is no provision for specifying a different precision. String data supports the full 8-bit space. Only 0x00 is reserved for string ending. This makes them ideal for storing UTF-8 encoded strings, ISO 8859-1, and plain ASCII. Both section keys and value keys are of this type. For data types not represented by the simple data types provided, use data handles containing your custom data. void\* unstructured data allows you to store any kind of data. You must pass in a size in bytes along with the data.

When calling any of the functions to retrieve the value of a key, if a given key is not found, the default value is both written to the blob and returned as the value; if no default is

provided, a blank value will be written and returned.

Note that this data is stored in the application’s preferences, not in the project. As of 6.5, there is no way to store opaque AEGP-generated data in an After Effects project.

After Effects can handle plug-ins which change the preferences during their application; it checks the in-RAM copy of the prefs before acting upon pref-able settings, rather than

relying on the saved prefs. It’s like we *planned* this, or something!

**TABLE 86: AEGP\_PERSISTENTDATASUITE4**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetApplicationBlob | Obtains the handle to all persistent application data. Modifying this will modify the application.  The AEGP\_PersistentType parameter is new in CC, and should be set to one of the following:  AEGP\_PersistentType\_MACHINE\_SPECIFIC, AEGP\_PersistentType\_MACHINE\_INDEPENDENT, AEGP\_PersistentType\_MACHINE\_INDEPENDENT\_RENDER, AEGP\_PersistentType\_MACHINE\_INDEPENDENT\_OUTPUT, AEGP\_PersistentType\_MACHINE\_INDEPENDENT\_  -COMPOSITION,  AEGP\_PersistentType\_MACHINE\_SPECIFIC\_TEXT, AEGP\_PersistentType\_MACHINE\_SPECIFIC\_PAINT  AEGP\_GetApplicationBlob( AEGP\_PersistentType blob\_type, AEGP\_PersistentBlobH \*blobPH); |
| AEGP\_GetNumSections | Obtains the number of sections in the application blob.  AEGP\_GetNumSections( AEGP\_PersistentBlobH blobH,  A\_long \*num\_sectionPL); |
| AEGP\_GetSectionKeyByIndex | Obtains the key at the given index.  AEGP\_GetSectionKeyByIndex( AEGP\_PersistentBlobH blobH,  A\_long section\_index,  A\_long max\_section\_size,  A\_char \*section\_keyZ); |
| AEGP\_DoesKeyExist | Returns whether or not a given key/value pair exists with the blob.  AEGP\_DoesKeyExist( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_Boolean \*existsPB); |
| AEGP\_GetNumKeys | Retrieves the number of value keys in the section.  AEGP\_GetNumKeys( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ, A\_long \*num\_keysPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetValueKeyByIndex | Retrieves the value of the indexed key.  AEGP\_GetValueKeyByIndex( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  A\_long key\_index,  A\_long max\_key\_size,  A\_char \*value\_keyZ); |
| *For the functions below, if a given key is not found, the default value is both written to the blob and returned as the value; if no default is provided, a blank value will be written and returned.* | |
| AEGP\_GetDataHandle | Obtains the value associated with the given section’s key.  If using in-memory data structures, watch for endian issues.  AEGP\_GetDataHandle(  AEGP\_PluginID plugin\_id, AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ, AEGP\_MemHandle defaultH0, AEGP\_MemHandle \*valuePH); |
| AEGP\_GetData | Obtains the data located at a given section’s value.  AEGP\_GetData( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_u\_long data\_sizeLu,  const void \*defaultPV0,  void \*bufPV); |
| AEGP\_GetString | Obtains the string for a given section key’s value (and indicates its length in actual\_szLu0).  AEGP\_GetString( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  const A\_char \*defaultZ0,  A\_u\_long buf\_sizeLu,  char \*bufZ,  A\_u\_long \*actual\_szLu0); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLong | Obtains the A\_long associated with a given section key’s value.  AEGP\_GetLong( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_long defaultL,  A\_long \*valuePL); |
| AEGP\_GetFpLong | Obtains the A\_FpLong associated with a given section key’s value.  AEGP\_GetFpLong( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_FpLong defaultF,  A\_FpLong \*valuePF); |
| AEGP\_GetTime | New in CC. Obtains the A\_Time associated with a given section key’s value.  AEGP\_GetTime( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  const A\_Time \*defaultPT0,  A\_Time \*valuePT); |
| AEGP\_GetARGB | New in CC. Obtains the PF\_PixelFloat associated with a given section key’s value.  AEGP\_GetARGB(  AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ, const PF\_PixelFloat \*defaultP0, PF\_PixelFloat \*valueP); |
| AEGP\_SetDataHandle | Sets the given section key’s value to the handle passed in.  AEGP\_SetDataHandle( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ, const AEGP\_MemHandle valueH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetData | Sets the given section key’s value to the data contained in  dataPV.  AEGP\_SetData( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_u\_long data\_sizeLu,  const void \*dataPV); |
| AEGP\_SetString | Sets the given section key’s string to strZ.  AEGP\_SetString( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  const A\_char \*strZ); |
| AEGP\_SetLong | Sets the given section key’s value to valueL.  AEGP\_SetLong( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_long valueL); |
| AEGP\_SetFpLong | Sets the given section key’s value to valueF.  AEGP\_SetFpLong( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_FpLong valueF); |
| AEGP\_SetTime | New in CC. Sets the given section key’s value to  valuePT.  AEGP\_SetTime( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  A\_Time \*valuePT); |
| AEGP\_SetARGB | New in CC. Sets the given section key’s value to  valueP.  AEGP\_SetARGB(  AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ,  PF\_PixelFloat \*valueP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_DeleteEntry | Removes the given section’s value from the blob.  AEGP\_DeleteEntry( AEGP\_PersistentBlobH blobH,  const A\_char \*section\_keyZ,  const A\_char \*value\_keyZ); |
| AEGP\_GetPrefsDirectory | Get the path to the folder containing After Effects’ preference file. The path is a handle to a NULL-  terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetPrefsDirectory)( AEGP\_MemHandle \*unicode\_pathPH); |

**COLOR MANAGEMENT**

We’ve provided a function so AEGPs can obtain information on After Effects’ current color management settings.

**TABLE 87: AEGP\_COLORSETTINGSSUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetBlendingTables | Retrieves the current opaque PF\_EffectBlendingTables, for use with AEGP\_TransferRect.  AEGP\_GetBlendingTables(  PR\_RenderContextH  render\_contextH, PF\_EffectBlendingTables  \*blending\_tables); |
| AEGP\_DoesViewHaveColor- SpaceXform | Returns whether there is a colorspace transform applied to the current item view.  AEGP\_DoesViewHaveColorSpaceXform( AEGP\_ItemViewP viewP, A\_Boolean \*has\_xformPB); |
| AEGP\_XformWorkingToViewCol- orSpace | Changes the view colorspace of the source to be the working colorspace of the destination. Source and destination can be the same.  AEGP\_XformWorkingToViewColorSpace( AEGP\_ItemViewP viewP, AEGP\_WorldH srcH,  AEGP\_WorldH dstH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewWorkingSpace ColorProfile | Retrieves the opaque current working space ICC profile. Must be disposed. The “New” in the name does not indicate that you’re making up a new profile; rather, it’s part of our function naming standard; anything with “New” in the name allocates something which the caller must dispose.  AEGP\_GetNewWorkingSpaceColorProfile( AEGP\_PluginID aegp\_plugin\_id, AEGP\_MemHandle \*icc\_profPH); |
| AEGP\_GetNewColorProfileFrom ICCProfile | Retrieves a new AEGP\_ColorProfileP from After Effects, representing the specified ICC profile. The  caller must dispose of the returned AEGP\_ColorProfileP using [AEGP\_DisposeColorProfile()](#_bookmark679).  AEGP\_GetNewColorProfile FromICCProfile(  AEGP\_PluginID aegp\_plugin\_id, A\_long icc\_sizeL,  const void \*icc\_dataPV, AEGP\_ColorProfileP \*profilePP); |
| AEGP\_GetNewICCProfileFrom ColorProfile | Retrieves a new ICC profile (stored in an AEGP\_MemHandle) representing the specified color profile. Returned AEGP\_MemHandle must be disposed by the caller.  AEGP\_GetNewICCProfile FromColorProfile(  AEGP\_PluginID plugin\_id,  AEGP\_ConstColorProfileP  profileP, AEGP\_MemHandle \*profilePH); |
| AEGP\_GetNewColorProfile Description | Returns a textual description of the specified color profile. Text will be a null-terminated UTF16 string, which must be disposed by the caller.  AEGP\_GetNewColorProfileDescription( AEGP\_PluginID aegp\_plugin\_id,  AEGP\_ConstColorProfileP  profileP, AEGP\_MemHandle \*unicode\_descPH); |
| AEGP\_DisposeColorProfile | Disposes of a color profile, obtained using other functions in this suite.  AEGP\_DisposeColorProfile( AEGP\_ColorProfileP profileP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetColorProfileApproxi- mateGamma | Returns a floating point number approximating the gamma setting used by the specified color profile.  AEGP\_GetColorProfileApproximateGamma(  AEGP\_ConstColorProfileP  profileP,  A\_FpShort \*approx\_gammaP); |
| AEGP\_IsRGBColorProfile | Returns whether the specified color profile is RGB.  AEGP\_IsRGBColorProfile(  AEGP\_ConstColorProfileP  profileP,  A\_Boolean \*is\_rgbPB); |

**RENDER SUITES**

Since we introduced the AEGP API, we’ve been asked to provide functions for retrieving rendered frames. These function suites allows you to do just that. First, specify what you want rendered in the [AEGP\_RenderOptionsSuite](#_bookmark681) or [AEGP\_LayerRenderOptionsSuite](#_bookmark682). Then do the rendering with [AEGP\_RenderSuite](#_bookmark684).

**TABLE 88: AEGP\_RENDEROPTIONSSUITE4**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewFromItem | Returns the AEGP\_RenderOptionsH associated with a given  AEGP\_ItemH. If there are no options yet specified, After Effects  passes back an AEGP\_RenderOptionsH with render time set to 0, time step set to the current frame duration, field render set to PF\_Field\_FRAME, and the depth set to the highest resolution  specified within the item.  AEGP\_NewFromItem(  AEGP\_PluginID plugin\_id,  AEGP\_ItemH itemH, AEGP\_RenderOptionsH \*optionsPH); |
| AEGP\_Duplicate | Duplicates an AEGP\_RenderOptionsH into copyPH.  AEGP\_Duplicate(  AEGP\_PluginID plugin\_id, AEGP\_RenderOptionsH optionsH, AEGP\_RenderOptionsH \*copyPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_Dispose | Deletes an AEGP\_RenderOptionsH.  AEGP\_Dispose(  AEGP\_RenderOptionsH optionsH); |
| AEGP\_SetTime | Sets the render time of an AEGP\_RenderOptionsH.  AEGP\_SetTime(  AEGP\_RenderOptionsH optionsH, A\_Time time); |
| AEGP\_GetTime | Retrieves the render time of the given AEGP\_RenderOptionsH.  AEGP\_GetTime(  AEGP\_RenderOptionsH optionsH, A\_Time \*timeP); |
| AEGP\_SetTimeStep | Specifies the time step (duration of a frame) for the referenced  AEGP\_RenderOptionsH.  AEGP\_SetTimeStep(  AEGP\_RenderOptionsH optionsH, A\_Time time\_step); |
| AEGP\_GetTimeStep | Retrieves the time step (duration of a frame) for the given  AEGP\_RenderOptionsH.  AEGP\_GetTimeStep(  AEGP\_RenderOptionsH optionsH, A\_Time \*timePT); |
| AEGP\_SetFieldRender | Specifies the field settings for the given AEGP\_RenderOptionsH.  AEGP\_SetFieldRender( AEGP\_RenderOptionsH optionsH,  PF\_Field field\_render); |
| AEGP\_GetFieldRender | Retrieves the field settings for the given AEGP\_RenderOptionsH.  AEGP\_GetFieldRender( AEGP\_RenderOptionsH optionsH,  PF\_Field \*field\_renderP); |
| AEGP\_SetWorldType | Specifies the AEGP\_WorldType of the output of a given  AEGP\_RenderOptionsH.  AEGP\_SetWorldType( AEGP\_RenderOptionsH optionsH, AEGP\_WorldType type);  AEGP\_WorldType will be either AEGP\_WorldType\_8 or  AEGP\_WorldType\_16 |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetWorldType | Retrieves the AEGP\_WorldType of the given  AEGP\_RenderOptionsH.  AEGP\_GetWorldType( AEGP\_RenderOptionsH optionsH, AEGP\_WorldType \*typeP); |
| AEGP\_SetDownsample Factor | Specifies the downsample factor (with independent horizontal and vertical settings) for the given AEGP\_RenderOptionsH.  AEGP\_SetDownsampleFactor( AEGP\_RenderOptionsH optionsH, A\_short x,  A\_short y); |
| AEGP\_GetDownsample Factor | Retrieves the downsample factor for the given  AEGP\_RenderOptionsH.  AEGP\_GetDownsampleFactor( AEGP\_RenderOptionsH optionsH, A\_short \*xP,  A\_short \*yP); |
| AEGP\_SetRegionOf Interest | Specifies the region of interest sub-rectangle for the given  AEGP\_RenderOptionsH.  AEGP\_SetRegionOfInterest( AEGP\_RenderOptionsH optionsH, const A\_LRect \*roiP) |
| AEGP\_GetRegionOf Interest | Retrieves the region of interest sub-rectangle for the given  AEGP\_RenderOptionsH.  AEGP\_GetRegionOfInterest( AEGP\_RenderOptionsH optionsH, A\_LRect \*roiP); |
| AEGP\_SetMatteMode | Specifies the AEGP\_MatteMode for the given  AEGP\_RenderOptionsH.  AEGP\_SetMatteMode( AEGP\_RenderOptionsH optionsH, AEGP\_MatteMode mode);  AEGP\_MatteMode will be one of the following:  AEGP\_MatteMode\_STRAIGHT AEGP\_MatteMode\_PREMUL\_BLACK AEGP\_MatteMode\_PREMUL\_BG\_COLOR |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetMatteMode | Retrieves the AEGP\_MatteMode for the given  AEGP\_RenderOptionsH.  AEGP\_GetMatteMode( AEGP\_RenderOptionsH optionsH, AEGP\_MatteMode \*modeP); |
| AEGP\_GetChannelOrder | Gets the AEGP\_ChannelOrder for the given AEGP\_RenderOptionsH. AEGP\_ChannelOrder will be either AEGP\_ChannelOrder\_ARGB or AEGP\_ChannelOrder\_BGRA.  AEGP\_GetChannelOrder( AEGP\_RenderOptionsH optionsH, AEGP\_ChannelOrder \*orderP);  Factoid: this was added to facilitate live linking with Premiere Pro. |
| AEGP\_SetChannelOrder | Sets the AEGP\_ChannelOrder of the AEGP\_RenderOptionsH.  AEGP\_SetChannelOrder( AEGP\_RenderOptionsH optionsH, AEGP\_ChannelOrder order); |
| AEGP\_GetRenderGuide Layers | Passes back a boolean that is true if the render guide layers setting is on.  AEGP\_GetRenderGuideLayers)( AEGP\_RenderOptionsH optionsH, A\_Boolean \*will\_renderPB); |
| AEGP\_SetRenderGuide Layers | Specify whether or not to render guide layers.  AEGP\_SetRenderGuideLayers)( AEGP\_RenderOptionsH optionsH, A\_Boolean render\_themB); |
| AEGP\_GetRenderQuality | Get the render quality of the render queue item. Quality can be either  AEGP\_ItemQuality\_DRAFT or AEGP\_ItemQuality\_BEST.  AEGP\_GetRenderQuality)( AEGP\_RenderOptionsH optionsH, AEGP\_ItemQuality \*qualityP); |
| AEGP\_SetRenderQuality | Set the render quality of the render queue item.  AEGP\_GetRenderQuality)( AEGP\_RenderOptionsH optionsH, AEGP\_ItemQuality quality); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewFromLayer | Returns the AEGP\_LayerRenderOptionsH associated with a  given AEGP\_LayerH. Render time is set to the layer's current time, time step is set to layer's frame duration, ROI to the layer's nominal bounds, and EffectsToRender to "all". optionsPH must be disposed by calling code.  AEGP\_NewFromLayer(  AEGP\_PluginID plugin\_id,  AEGP\_LayerH layerH, AEGP\_LayerRenderOptionsH \*optionsPH); |
| AEGP\_NewFromUpstrea- mOfEffect | Returns the AEGP\_LayerRenderOptionsH from the layer  associated with a given AEGP\_EffectRefH. Render time is set to the layer's current time, time step is set to layer's frame duration, ROI to the layer's nominal bounds, and EffectsToRender to the index of effectH. optionsPH must be disposed by calling code.  AEGP\_NewFromUpstreamOfEffect( AEGP\_PluginID plugin\_id,  AEGP\_EffectRefH effectH, AEGP\_LayerRenderOptionsH \*optionsPH); |
| AEGP\_Duplicate | Duplicates an AEGP\_LayerRenderOptionsH into copyPH.  AEGP\_Duplicate(  AEGP\_PluginID plugin\_id, AEGP\_LayerRenderOptionsH optionsH, AEGP\_LayerRenderOptionsH \*copyPH); |
| AEGP\_Dispose | Deletes an AEGP\_LayerRenderOptionsH.  AEGP\_Dispose(  AEGP\_LayerRenderOptionsH optionsH); |
| AEGP\_SetTime | Sets the render time of an AEGP\_LayerRenderOptionsH.  AEGP\_SetTime(  AEGP\_LayerRenderOptionsH optionsH, A\_Time time); |
| AEGP\_GetTime | Retrieves the render time of the given  AEGP\_LayerRenderOptionsH.  AEGP\_GetTime(  AEGP\_LayerRenderOptionsH optionsH, A\_Time \*timeP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetTimeStep | Specifies the time step (duration of a frame) for the referenced  AEGP\_LayerRenderOptionsH.  AEGP\_SetTimeStep( AEGP\_LayerRenderOptionsH optionsH, A\_Time time\_step); |
| AEGP\_GetTimeStep | Retrieves the time step (duration of a frame) for the given  AEGP\_LayerRenderOptionsH.  AEGP\_GetTimeStep( AEGP\_LayerRenderOptionsH optionsH, A\_Time \*timePT); |
| AEGP\_SetWorldType | Specifies the AEGP\_WorldType of the output of a given  AEGP\_LayerRenderOptionsH.  AEGP\_SetWorldType( AEGP\_LayerRenderOptionsH optionsH, AEGP\_WorldType type);  AEGP\_WorldType will be either AEGP\_WorldType\_8 or  AEGP\_WorldType\_16 |
| AEGP\_GetWorldType | Retrieves the AEGP\_WorldType of the given  AEGP\_LayerRenderOptionsH.  AEGP\_GetWorldType( AEGP\_LayerRenderOptionsH optionsH, AEGP\_WorldType \*typeP); |
| AEGP\_SetDownsample Factor | Specifies the downsample factor (with independent horizontal and vertical settings) for the given AEGP\_LayerRenderOptionsH.  AEGP\_SetDownsampleFactor( AEGP\_LayerRenderOptionsH optionsH, A\_short x,  A\_short y); |
| AEGP\_GetDownsample Factor | Retrieves the downsample factor for the given  AEGP\_LayerRenderOptionsH.  AEGP\_GetDownsampleFactor( AEGP\_LayerRenderOptionsH optionsH, A\_short \*xP,  A\_short \*yP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetMatteMode | Specifies the AEGP\_MatteMode for the given  AEGP\_LayerRenderOptionsH.  AEGP\_SetMatteMode( AEGP\_LayerRenderOptionsH optionsH, AEGP\_MatteMode mode);  AEGP\_MatteMode will be one of the following:  AEGP\_MatteMode\_STRAIGHT AEGP\_MatteMode\_PREMUL\_BLACK AEGP\_MatteMode\_PREMUL\_BG\_COLOR |
| AEGP\_GetMatteMode | Retrieves the AEGP\_MatteMode for the given  AEGP\_LayerRenderOptionsH.  AEGP\_GetMatteMode( AEGP\_LayerRenderOptionsH optionsH, AEGP\_MatteMode \*modeP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RenderAndCheck- outFrame | Retrieves an AEGP\_FrameReceiptH (not the actual pixels) for the frame requested. Check in this receipt using AEGP\_CheckinFrame to release memory.  Create the AEGP\_RenderOptionsH using the  [AEGP\_RenderOptionsSuite](#_bookmark681).  Optionally, the AEGP can pass a function to be called by After Effects if the user cancels the current render, as well as a refcon (constant reference to opaque data) for use during that  function.  AEGP\_RenderAndCheckoutFrame(  AEGP\_RenderOptionsH  optionsH,  AEGP\_RenderSuiteCheckForCancel  cancel\_functionP0,  AEGP\_CancelRefcon  cancel\_function\_refconP0, AEGP\_FrameReceiptH  \*receiptPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RenderAndCheck- outLayerFrame | New in CC 2014. This allows frame checkout of a layer with effects applied at non-render time. This is useful for an  operation that requires the frame, for example, when a button is clicked and it is acceptable to wait for a moment while it is  rendering. Note: Since it is not asynchronous, it will not solve the general problem where custom UI needs to draw based on the frame.  Retrieves an AEGP\_FrameReceiptH (not the actual pixels) for the layer frame requested. Check in this receipt using AEGP\_CheckinFrame to release memory.  Create the AEGP\_LayerRenderOptionsH using AEGP\_NewFromUpstreamOfEffect(), in the [AEGP\_LayerRenderOptionsSuite](#_bookmark682).  You can actually use AEGP\_NewFromLayer() to get other layer param's layers with their effects applied. However, be careful. If you do it in your effect A, and there's an effect B on  the other layer that does the same thing during rendering, you'd create an infinite loop. If you're not doing it for render  purposes then it could be okay.  Optionally, the AEGP can pass a function to be called by After Effects if the user cancels the current render, as well as a refcon (constant reference to opaque data) for use during that  function.  AEGP\_RenderAndCheckoutLayerFrame(  AEGP\_LayerRenderOptionsH  optionsH,  A\_Boolean render\_plain\_layer\_frameB,  AEGP\_RenderSuiteCheckForCancel  cancel\_functionP0,  AEGP\_CancelRefcon  cancel\_function\_refconP0, AEGP\_FrameReceiptH  \*receiptPH); |
| AEGP\_CheckinFrame | Call this function as soon as your AEGP is done accessing the frame. After Effects makes caching decisions based on which  frames are checked out, so don’t hog them!  AEGP\_CheckinFrame(  AEGP\_FrameReceiptH receiptH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetReceiptWorld | Retrieves the pixels (AEGP\_WorldH) associated with the referenced AEGP\_FrameReceiptH.  AEGP\_GetReceiptWorld( AEGP\_FrameReceiptH receiptH, AEGP\_WorldH \*worldPH); |
| AEGP\_GetRenderedRegion | Retrieves an A\_LRect containing the region of the AEGP\_FrameReceiptH‘s AEGP\_WorldH that has already been rendered. Remember that it’s possible for only those  portions of an image that have been changed to be rendered, so it’s important to be able to check whether or not that includes the portion you need.  AEGP\_GetRenderedRegion( AEGP\_FrameReceiptH receiptH, A\_LRect \*regionP); |
| AEGP\_IsRenderedFrame Sufficient | Given two sets of AEGP\_RenderOptionsH, After Effects will return TRUE if the already-rendered pixels are still valid for the proposed AEGP\_RenderOptionsH.  AEGP\_IsRenderedFrameSufficient( AEGP\_RenderOptionsH rendered\_optionsH, AEGP\_RenderOptionsH proposed\_optionsH, A\_Boolean \*is\_sufficientPB); |
| AEGP\_RenderNewItem SoundData | Obtains an AEGP\_ItemH’s audio at the given time, of the given duration, in the given format. The plug-in must dispose of the returned AEGP\_SoundDataH (which may be NULL if no audio is available).  AEGP\_RenderNewItemSoundData( AEGP\_ItemH itemH,  const A\_Time \*start\_timePT,  const A\_Time \*durationPT, const AEGP\_SoundDataFormat  \*formatP,  AEGP\_SoundDataH \*new\_dataPH);  NOTE: This function, if called as part of AEGP\_ItemSuite2, provides a render interruptible using mouse clicks, unlike the version published here in AEGP\_RenderSuite. |
| AEGP\_GetCurrent Timestamp | Retrieves the current AEGP\_TimeStamp of the project. The AEGP\_TimeStamp is updated whenever an item is touched in a way that affects rendering.  AEGP\_GetCurrentTimestamp( AEGP\_TimeStamp \*time\_stampP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_HasItemChanged SinceTimestamp | Returns whether the video of an AEGP\_ItemH has changed  since the given AEGP\_TimeStamp. Note: this does not track changes in audio.  AEGP\_HasItemChangedSinceTimestamp( AEGP\_ItemH itemH,  const A\_Time \*start\_timeP,  const A\_Time \*durationP, const AEGP\_TimeStamp \*time\_stampP, A\_Boolean \*changedPB); |
| AEGP\_IsItemWorthwhile ToRender | Returns whether this frame would be worth rendering  externally and checking in to the cache. A speculative renderer should check this twice: before sending the frame out to render and when it is complete, before calling [AEGP\_NewPlatformWorld()](#_bookmark692) and checking in. This  function is to be used with [AEGP\_HasItemChangedSinceTimestamp()](#_bookmark688), not alone.  AEGP\_IsItemWorthwhileToRender( AEGP\_RenderOptionsH roH,  const AEGP\_TimeStamp \*time\_stampP, A\_Boolean \*worthwhilePB); |
| AEGP\_Checkin RenderedFrame | Provide a rendered frame (AEGP\_PlatformWorldH) to After Effects, which adopts it. ticksL is the approximate time required to render the frame.  AEGP\_CheckinRenderedFrame( AEGP\_RenderOptionsH roH,  const AEGP\_TimeStamp\* time\_stampP, A\_u\_long ticksL,  AEGP\_PlatformWorldH imageH); |
| AEGP\_GetReceiptGuid | New in CS6. Retrieve a GUID for a rendered frame. The memory handle passed back must be disposed.  AEGP\_GetReceiptGuid( AEGP\_FrameReceiptH receiptH, AEGP\_MemHandle \*guidMH) |

AEGP\_Worlds are the common format used throughout the AEGP APIs to describe frames of pixels.

**TABLE 91: AEGP\_WORLDSUITE3**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_New | Returns an allocated, initialized AEGP\_WorldH.  AEGP\_New(  AEGP\_PluginID plugin\_id,  AEGP\_WorldType type,  A\_long widthL,  A\_long heightL,  AEGP\_WorldH \*worldPH); |
| AEGP\_Dispose | Disposes of an AEGP\_WorldH. Use this on every world you allocate.  AEGP\_Dispose(  AEGP\_WorldH worldH); |
| AEGP\_GetType | Returns the type of a given AEGP\_WorldH.  AEGP\_GetType(  AEGP\_WorldH worldH, AEGP\_WorldType \*\*typeP);  AEGP\_WorldType will be one of the following:  AEGP\_WorldType\_8, AEGP\_WorldType\_16, AEGP\_WorldType\_32 |
| AEGP\_GetSize | Returns the width and height of the given  AEGP\_WorldH.  AEGP\_GetSize(  AEGP\_WorldH worldH,  A\_long \*widthPL,  A\_long \*heightPL); |
| AEGP\_GetRowBytes | Returns the rowbytes for the given AEGP\_WorldH.  AEGP\_GetRowBytes(  AEGP\_WorldH worldH,  A\_u\_long \*row\_bytesPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetBaseAddr8 | Returns the base address of the AEGP\_WorldH for use in pixel iteration functions. Will return an error if used on a non-8bpc world.  AEGP\_GetBaseAddr8(  AEGP\_WorldH worldH,  PF\_Pixel8 \*\*base\_addrP); |
| AEGP\_GetBaseAddr16 | Returns the base address of the AEGP\_WorldH for use in pixel iteration functions. Will return an error if used on a non-16bpc world.  AEGP\_GetBaseAddr16( AEGP\_WorldH worldH,  PF\_Pixel16 \*\*base\_addrP); |
| AEGP\_GetBaseAddr32 | Returns the base address of the AEGP\_WorldH for use in pixel iteration functions. Will return an error if used on a non-32bpc world.  AEGP\_GetBaseAddr32( AEGP\_WorldH worldH,  PF\_PixelFloat \*\*base\_addrP); |
| AEGP\_FillOutPFEffectWorld | Populates and returns a PF\_EffectWorld  representing the given AEGP\_WorldH, for use with numerous pixel processing callbacks.  NOTE: This does not give your plug-in ownership of the world referenced; destroy the source AEGP\_WorldH  only if you allocated it. it just fills out the provided  PF\_EffectWorld to point to the same pixel buffer.  AEGP\_FillOutPFEffectWorld( AEGP\_WorldH worldH, PF\_EffectWorld \*pf\_worldP); |
| AEGP\_FastBlur | Performs a fast blur on a given AEGP\_WorldH.  AEGP\_FastBlur(  A\_FpLong radiusF,  PF\_ModeFlags mode,  PF\_Quality quality,  AEGP\_WorldH worldH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewPlatformWorld | Creates a new AEGP\_PlatformWorldH (a pixel world native to the execution platform).  AEGP\_NewPlatformWorld( AEGP\_PluginID plugin\_id,  AEGP\_WorldType type,  A\_long widthL,  A\_long heightL,  AEGP\_PlatformWorldH \*worldPH); |
| AEGP\_DisposePlatformWorld | Disposes of an AEGP\_PlatformWorldH.  AEGP\_DisposePlatformWorld( AEGP\_PlatformWorldH worldH); |
| AEGP\_NewReferenceFrom PlatformWorld | Retrieves an AEGP\_WorldH referring to the given  AEGP\_PlatformWorldH. NOTE: This doesn’t  allocate a new world, it simply provides a reference to an existing one.  AEGP\_NewReferenceFromPlatformWorld( AEGP\_PluginID plugin\_id, AEGP\_PlatformWorldH plat\_worldH, AEGP\_WorldH \*worldPH); |

Use the AEGP\_CompositeSuite to copy pixel worlds, operate on track mattes, and apply transfer functions.

**TABLE 92: AEGP\_COMPOSITESUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_ClearAlphaExceptRect | For the given PF\_EffectWorld, sets the alpha to fully transparent except for the specified rectangle.  AEGP\_ClearAlphaExceptRect(  A\_Rect \*clipped\_dst\_rectPR,  PF\_EffectWorld \*dstP); |
| AEGP\_PrepTrackMatte | Mattes the pixels in a PF\_EffectWorld with the PF\_Pixel  described in src\_masks, putting the output into an array of pixels dst\_mask. NOTE: Unlike most of the other pixel mangling functions provided by After Effects, this one doesn’t  take PF\_EffectWorld arguments; rather, you can simply pass the data pointer from within the PF\_EffectWorld. This can be confusing, but as a bonus, the function pads output  appropriately so that num\_pix pixels are always output.  AEGP\_PrepTrackMatte(  A\_long num\_pix,  A\_Boolean deepB, const PF\_Pixel \*src\_mask, PF\_MaskFlags mask\_flags,  PF\_Pixel \*dst\_mask); |
| AEGP\_TransferRect | Blends two PF\_EffectWorlds using a transfer mode, with an optional mask. Pass NULL for the blend\_tablesP0  parameter to perform blending in the current working color space.  AEGP\_TransferRect(  PF\_Quality quality,  PF\_ModeFlags m\_flags,  PF\_Field field,  const A\_Rect \*src\_rec, const PF\_EffectWorld \*src\_world, const PF\_CompositeMode \*comp\_mode,  PF\_EffectBlendingTables blend\_tablesP0, const PF\_MaskWorld \*mask\_world0, A\_long dest\_x,  A\_long dest\_y,  PF\_EffectWorld \*dst\_world); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_CopyBits\_LQ | Copies a rectangle of pixels (pass a NULL rectangle to get all  pixels) from one PF\_EffectWorld to another, at low quality.  AEGP\_CopyBits\_LQ(  PF\_EffectWorld \*src\_worldP,  A\_Rect \*src\_r,  A\_Rect \*dst\_r,  PF\_EffectWorld \*dst\_worldP); |
| AEGP\_CopyBits\_HQ\_Straight | Copies a rectangle of pixels (pass a NULL rectangle to get all  pixels) from one PF\_EffectWorld to another, at high quality, with a straight alpha channel.  AEGP\_CopyBits\_HQ\_Straight( PF\_EffectWorld \*src,  A\_Rect \*src\_r,  A\_Rect \*dst\_r,  PF\_EffectWorld \*dst); |
| AEGP\_CopyBits\_HQ\_Premul | Copies a rectangle of pixels (pass a NULL rectangle to get all  pixels) from one PF\_EffectWorld to another, at high quality, premultiplying the alpha channel.  AEGP\_CopyBits\_HQ\_Premul( PF\_EffectWorld \*src,  A\_Rect \*src\_r,  A\_Rect \*dst\_r,  PF\_EffectWorld \*dst); |

AEGP\_SoundDataSuite allows AEGPs to obtain and manipulate the audio associated with compositions and footage items. Audio-only items may be added to the render queue using AEGP\_RenderNewItemSoundData().

**TABLE 93: AEGP\_SOUNDDATASUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_NewSoundData | Creates a new AEGP\_SoundDataH, of which the plug-in must dispose.  AEGP\_NewSoundData(  const AEGP\_SoundDataFormat  \*formatP, AEGP\_SoundDataH \*new\_dataPH); |
| AEGP\_DisposeSoundData | Frees an AEGP\_SoundDataH.  AEGP\_DisposeSoundData( AEGP\_SoundDataH sound\_dataH); |
| AEGP\_GetSoundDataFormat | Obtains information about the format of a given  AEGP\_SoundDataH.  AEGP\_GetSoundDataFormat( AEGP\_SoundDataH soundH, AEGP\_SoundDataFormat \*formatP); |
| AEGP\_LockSoundDataSamples | Locks the AEGP\_SoundDataH in memory.  AEGP\_LockSoundDataSamples( AEGP\_SoundDataH soundH, void \*\*samples); |
| AEGP\_UnlockSoundDataSamples | Unlocks an AEGP\_SoundDataH.  AEGP\_UnlockSoundDataSamples( AEGP\_SoundDataH soundH); |
| AEGP\_GetNumSamples | Obtains the number of samples in the given  AEGP\_SoundDataH.  AEGP\_GetNumSamples( AEGP\_SoundDataH soundH,  A\_long \*numsamplesPL); |

**AUDIO SETTINGS**

Audio render settings are represented using the AEGP\_SoundDataFormat.

struct AEGP\_SoundDataFormat {

A\_FpLong sample\_rateF; AEGP\_SoundEncoding encoding;

A\_long bytes\_per\_sampleL;

A\_long num\_channelsL;// 1 for mono, 2 for stereo

}AEGP\_SoundDataFormat;

bytes\_per\_sampleL is always either 1, 2, or 4, and is ignored if float encoding is specified.

AEGP\_SoundEncoding is one of the following: AEGP\_SoundEncoding\_UNSIGNED\_PCM AEGP\_SoundEncoding\_SIGNED\_PCM AEGP\_SoundEncoding\_FLOAT

**RENDER QUEUE SUITE**

This suite allows AEGPs to add items the to render queue (using default options), and control the basic state of the render queue.

**TABLE 94: AEGP\_RENDERQUEUESUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_AddCompToRenderQueue | Adds a composition to the render queue, using default options.  AEGP\_AddCompToRenderQueue( AEGP\_CompH compH,  const A\_char\* pathZ); |
| AEGP\_SetRenderQueueState | Sets the render queue to one of three valid states. It is not possible to go from stopped to paused.  AEGP\_SetRenderQueueState( AEGP\_RenderQueueState state);  AEGP\_RenderQueueState\_STOPPED AEGP\_RenderQueueState\_PAUSED AEGP\_RenderQueueState\_RENDERING |
| AEGP\_GetRenderQueueState | Obtains the current render queue state.  AEGP\_GetRenderQueueState( AEGP\_RenderQueueState \*stateP); |

Manipulate all aspects of render queue items using this suite.

**TABLE 95: AEGP\_RQITEMSUITE3**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNumRQItems | Returns the number of items currently in the render queue.  AEGP\_GetNumRQItems(  A\_long \*num\_itemsPL); |
| AEGP\_GetRQItemByIndex | Returns an AEGP\_RQItemRefH referencing the index’d item.  AEGP\_GetRQItemByIndex(  A\_long rq\_item\_index, AEGP\_RQItemRefH \*rq\_item\_refPH); |
| AEGP\_GetNextRQItem | Returns the next AEGP\_RQItemRefH, for iteration purposes. To get the first AEGP\_RQItemRefH, pass RQ\_ITEM\_INDEX\_NONE for the current\_rq\_itemH.  AEGP\_GetNextRQItem(  AEGP\_RQItemRefH current\_rq\_itemH, AEGP\_RQItemRefH \*next\_rq\_itemPH); |
| AEGP\_GetNumOutputModules ForRQItem | Returns the number of output modules applied to the given  AEGP\_RQItemRefH.  AEGP\_GetNumOutputModulesForRQItem( AEGP\_RQItemRefH rq\_itemH,  A\_long \*num\_outmodsPL); |
| AEGP\_GetRenderState | Returns TRUE if the AEGP\_RQItemRefH is set to render (once the user clicks the Render button).  AEGP\_GetRenderState( AEGP\_RQItemRefH rq\_itemH,  A\_Boolean \*will\_renderPB); |
| AEGP\_SetRenderState | Controls whether or not the AEGP\_RQItemRefH will render when the user next clicks the Render button. Returns an error if called during rendering. This function will return Err\_PARAMETER if you try to call while AEGP\_RenderQueueState isn’t AEGP\_RenderQueueState\_STOPPED, Err\_RANGE if you pass a status that is illegal in any case, and Err\_PARAMETER if you try to pass a status that doesn't make sense (like trying to  queue something for which there’s no output path)  AEGP\_SetRenderState( AEGP\_RQItemRefH rq\_itemH, A\_Boolean renderB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetStartedTime | Returns the time (in seconds, since 1904) that rendering began.  AEGP\_GetStartedTime( AEGP\_RQItemRefH rq\_itemH,  A\_Time \*started\_timePT); |
| AEGP\_GetElapsedTime | Returns the time elapsed since rendering began.  AEGP\_GetElapsedTime( AEGP\_RQItemRefH rq\_itemH,  A\_Time \*render\_timePT); |
| AEGP\_GetLogType | Returns the log type for the referenced AEGP\_RQItemRefH.  AEGP\_GetLogType(  AEGP\_RQItemRefH rq\_itemH,  AEGP\_LogType \*logtypeP); AEGP\_LogtType will have one of the following values:  AEGP\_LogType\_NONE AEGP\_LogType\_ERRORS\_ONLY AEGP\_LogType\_PLUS\_SETTINGS AEGP\_LogType\_PER\_FRAME\_INFO |
| AEGP\_SetLogType | Specifies the log type to be used with the referenced  AEGP\_RQItemRefH.  AEGP\_SetLogType(  AEGP\_RQItemRefH rq\_itemH,  AEGP\_LogType logtype); |
| AEGP\_RemoveOutputModule | Removes the specified output module from the referenced  AEGP\_RQItemRefH.  AEGP\_RemoveOutputModule( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH); |
| AEGP\_GetComment | Retrieves the comment associated with the referenced  AEGP\_RQItemRefH.  AEGP\_GetComment(  AEGP\_RQItemRefH rq\_itemH,  A\_char \*commentZ); |
| AEGP\_SetComment | Specifies the comment associated with the referenced  AEGP\_RQItemRefH.  AEGP\_SetComment(  AEGP\_RQItemRefH rq\_itemH,  const A\_char \*commentZ); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompFromRQItem | Retrieves the AEGP\_CompH associated with the  AEGP\_RQItemRefH.  AEGP\_GetCompFromRQItem( AEGP\_RQItemRefH rq\_itemH,  AEGP\_CompH \*compPH); |
| AEGP\_DeleteRQItem | Deletes the render queue item. Undoable.  AEGP\_DeleteRQItem(  AEGP\_RQItemRefH rq\_itemH); |

**RENDER QUEUE MONITOR SUITE**

New in CS6. This suite provides all the info a render queue manager needs to figure out what is happening at any point in a render.

**TABLE 96: AEGP\_RENDERQUEUEMONITORSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RegisterListener | Register a set of plug-in-defined functions to be called by the render queue. Use the refcon to pass in data that you want to use later on when your plug-in-defined functions in AEGP\_RQM\_FunctionBlock1 are called later. It may be set it to NULL if you don’t need it.  AEGP\_RegisterListener(  AEGP\_PluginID aegp\_plugin\_id, AEGP\_RQM\_Refcon aegp\_refconP, const AEGP\_RQM\_FunctionBlock1  \*fcn\_blockP);  The AEGP\_RQM\_FunctionBlock1 is defined as follows:  struct \_AEGP\_RQM\_FunctionBlock1 { A\_Err (\*AEGP\_RQM\_RenderJobStarted)(  AEGP\_RQM\_BasicData \*basic\_dataP, AEGP\_RQM\_SessionId jobid);  A\_Err (\*AEGP\_RQM\_RenderJobEnded)( AEGP\_RQM\_BasicData \*basic\_dataP, AEGP\_RQM\_SessionId jobid);  (AEGP\_RQM\_FunctionBlock1 definition continued on next page) |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RegisterListener (cont’d) | A\_Err (\*AEGP\_RQM\_RenderJobItemStarted)( AEGP\_RQM\_BasicData \*basic\_dataP, AEGP\_RQM\_SessionId jobid, AEGP\_RQM\_ItemId itemid);  A\_Err (\*AEGP\_RQM\_RenderJobItemUpdated)( AEGP\_RQM\_BasicData \*basic\_dataP, AEGP\_RQM\_SessionId jobid, AEGP\_RQM\_ItemId itemid, AEGP\_RQM\_FrameId frameid);  A\_Err (\*AEGP\_RQM\_RenderJobItemEnded)( AEGP\_RQM\_BasicData \*basic\_dataP, AEGP\_RQM\_SessionId jobid, AEGP\_RQM\_ItemId itemid, AEGP\_RQM\_FinishedStatus fstatus);  AEGP\_RQM\_FinishedStatus\_UNKNOWN, AEGP\_RQM\_FinishedStatus\_SUCCEEDED, AEGP\_RQM\_FinishedStatus\_ABORTED, AEGP\_RQM\_FinishedStatus\_ERRED  A\_Err (\*AEGP\_RQM\_RenderJobItemReportLog)( AEGP\_RQM\_BasicData \*basic\_dataP, AEGP\_RQM\_SessionId jobid, AEGP\_RQM\_ItemId itemid, A\_Boolean isError, AEGP\_MemHandle logbuf);  } AEGP\_RQM\_FunctionBlock1;  The AEGP\_RQM\_BasicData is defined below.  struct \_AEGP\_RQM\_BasicData { const struct SPBasicSuite  \*pica\_basicP;  A\_long aegp\_plug\_id; AEGP\_RQM\_Refcon aegp\_refconPV;  } AEGP\_RQM\_BasicData; |
| AEGP\_DeregisterListener | Deregister from the render queue.  AEGP\_DeregisterListener( AEGP\_PluginID aegp\_plugin\_id, AEGP\_RQM\_Refcon aegp\_refconP); |
| AEGP\_GetProjectName | Obtain the current project name. The project name is a handle to a NULL-terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetProjectName( AEGP\_RQM\_SessionId sessid,  AEGP\_MemHandle \*utf\_project\_namePH0); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetAppVersion | Obtain the app version. The app version is a handle to a NULL-terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetAppVersion( AEGP\_RQM\_SessionId sessid,  AEGP\_MemHandle \*utf\_app\_versionPH0); |
| AEGP\_GetNumJobItems | Obtain the number of job items.  AEGP\_GetNumJobItems( AEGP\_RQM\_SessionId sessid,  A\_long \*num\_jobitemsPL); |
| AEGP\_GetJobItemID | Get the job with the index specified.  AEGP\_GetJobItemID( AEGP\_RQM\_SessionId sessid, A\_long jobItemIndex, AEGP\_RQM\_ItemId \*jobItemID); |
| AEGP\_GetNumJobItemRenderSettings | Get the number of render settings for the job with the index specified.  AEGP\_GetNumJobItemRenderSettings( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long \*num\_settingsPL); |
| AEGP\_GetJobItemRenderSetting | Get a specific render setting of a specific job. The setting name and value are handles to NULL-terminated A\_UTF16Char strings, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetJobItemRenderSetting( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid, A\_long settingIndex,  AEGP\_MemHandle \*utf\_setting\_namePH0, AEGP\_MemHandle \*utf\_setting\_valuePH0); |
| AEGP\_GetNumJobItemOutputModules | Get the number of output modules for the job with the index specified.  AEGP\_GetNumJobItemOutputModules( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long \*num\_outputmodulesPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNumJobItemOutputModule Settings | Get the number of settings for the output module with the index specified.  AEGP\_GetNumJobItemOutputModuleSettings( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long outputModuleIndex,  A\_long \*num\_settingsPL); |
| AEGP\_GetJobItemOutputModule Setting | Get a specific setting of a job item output module. The  setting name and value are handles to NULL-terminated A\_UTF16Char strings, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetJobItemOutputModuleSetting( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long outputModuleIndex,  A\_long settingIndex, AEGP\_MemHandle \*utf\_setting\_namePH0, AEGP\_MemHandle \*utf\_setting\_valuePH0); |
| AEGP\_GetNumJobItemOutputModule Warnings | Get the number of output module warnings for a job item.  AEGP\_GetNumJobItemOutputModuleWarnings( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long outputModuleIndex,  A\_long \*num\_warningsPL); |
| AEGP\_GetJobItemOutputModule Warning | Get a specific warning of a specific output module for a specific job item. The warning value is a handle to  NULL-terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetJobItemOutputModuleWarning( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long outputModuleIndex,  A\_long warningIndex, AEGP\_MemHandle \*utf\_warning\_valuePH0); |
| AEGP\_GetNumJobItemFrame Properties | Get the number of properties for a job item frame.  AEGP\_GetNumJobItemFrameProperties( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid, AEGP\_RQM\_FrameId frameid,  A\_long \*num\_propertiesPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetJobItemFrameProperty | Get a specific property on a job item frame. The property name and values are handle to NULL-terminated A\_UTF16Char strings, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetJobItemFrameProperty( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid, AEGP\_RQM\_FrameId frameid, A\_long propertyIndex,  AEGP\_MemHandle \*utf\_property\_namePH0, AEGP\_MemHandle \*utf\_property\_valuePH0); |
| AEGP\_GetNumJobItemOutputModule Properties | Get the number of properties for a job item output module.  AEGP\_GetNumJobItemOutputModuleProperties( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long outputModuleIndex,  A\_long \*num\_propertiesPL); |
| AEGP\_GetJobItemOutputModule Property | Get a specific property off a job item output module. The property name and values are handle to NULL-  terminated A\_UTF16Char strings, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetJobItemOutputModuleProperty( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid,  A\_long outputModuleIndex,  A\_long propertyIndex, AEGP\_MemHandle \*utf\_property\_namePH0, AEGP\_MemHandle \*utf\_property\_valuePH0); |
| AEGP\_GetJobItemFrameThumbnail | Get a buffer with a JPEG-encoded thumbnail of the job item frame. Pass in the maximum width and height, and the actual dimensions will be passed back.  AEGP\_GetJobItemFrameThumbnail( AEGP\_RQM\_SessionId sessid, AEGP\_RQM\_ItemId itemid, AEGP\_RQM\_FrameId frameid, A\_long \*widthPL,  A\_long \*heightPL, AEGP\_MemHandle \*thumbnailPH0); |

Every item in the render queue has at least one output module specified. Use this suite to

query and control all aspects of the output modules attached to a given render item. You may also add and remove output modules. Factoid: For each frame rendered for a given render

item, the list of output modules is traversed. So, for frame 0, output module 0, then 1, then 2 are called.

**TABLE 97: AEGP\_OUTPUTMODULESUITE4**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetOutputModule ByIndex | Retrieves the indexed output module. NOTE: AEGP\_OutputModuleRefH is an opaque data type, and can’t be manipulated directly; you must use our accessor functions to  modify it.  AEGP\_GetOutputModuleByIndex( AEGP\_RQItemRefH rq\_itemH,  A\_long outmod\_indexL, AEGP\_OutputModuleRefH \*outmodPH); |
| AEGP\_GetEmbedOptions | Retrieves the embedding setting specified for the referenced  AEGP\_OutputModuleRefH.  AEGP\_GetEmbedOptions( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH,  AEGP\_EmbeddingType \*embed\_optionsP); AEGP\_EmbeddingType will be one of the following:  AEGP\_Embedding\_NOTHING AEGP\_Embedding\_LINK AEGP\_Embedding\_LINK\_AND\_COPY |
| AEGP\_SetEmbedOptions | Specifies the embedding setting for the referenced  AEGP\_OutputModuleRefH.  AEGP\_SetEmbedOptions( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH,  AEGP\_EmbeddingType embed\_options); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetPostRenderAction | Retrieves the post-render action setting for the referenced  AEGP\_OutputModuleRefH.  AEGP\_GetPostRenderAction( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, AEGP\_PostRenderAction \*actionP);  AEGP\_PostRenderAction will be one of the following:  AEGP\_PostRenderOptions\_IMPORT AEGP\_PostRenderOptions\_ IMPORT\_AND\_REPLACE\_USAGE  AEGP\_PostRenderOptions\_SET\_PROXY |
| AEGP\_SetPostRenderAction | Specifies the post-render action setting for the referenced  AEGP\_OutputModuleRefH.  AEGP\_SetPostRenderAction( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, AEGP\_PostRenderAction action); |
| AEGP\_GetEnabledOutputs | Retrieves which output types are enabled for the referenced  AEGP\_OutputModuleRefH.  AEGP\_GetEnabledOutputs( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, AEGP\_OutputTypes \*typesP);  AEGP\_OutputTypes will contain one or both of the following values:  AEGP\_OutputType\_VIDEO AEGP\_OutputType\_AUDIO  NOTE: These are flags, not an enumeration. |
| AEGP\_SetEnabledOutputs | Specifies which output types are enabled for the referenced  AEGP\_OutputModuleRefH.  AEGP\_SetEnabledOutputs( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH,  AEGP\_OutputTypes enabled\_types); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetOutputChannels | Retrieves which video channels are enabled for output in the referenced AEGP\_OutputModuleRefH.  AEGP\_GetOutputChannels( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH,  AEGP\_VideoChannels \*outchannelsP); AEGP\_VideoChannels will be one of the following:  AEGP\_VideoChannels\_RGB AEGP\_VideoChannels\_RGBA AEGP\_VideoChannels\_ALPHA |
| AEGP\_SetOutputChannels | Specifies which video channels are enabled for output in the referenced AEGP\_OutputModuleRefH.  AEGP\_SetOutputChannels( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, AEGP\_VideoChannels outchannels); |
| AEGP\_GetStretchInfo | Retrieves the stretch information enabled for the referenced AEGP\_OutputModuleRefH;whether or not stretching is enabled, whether or not the frame aspect ratio is locked to the composition’s, and what quality setting is specified.  AEGP\_GetStretchInfo( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, A\_Boolean \*enabledPB, AEGP\_StretchQuality \*qualP, A\_Boolean \*lockedPB);  AEGP\_StretchQuality will be one of the following:  AEGP\_StretchQual\_LOW AEGP\_StretchQual\_HIGH |
| AEGP\_SetStretchInfo | Retrieves the stretch information enabled for the referenced  AEGP\_OutputModuleRefH.  AEGP\_SetStretchInfo( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH,  A\_Boolean is\_enabledB, AEGP\_StretchQuality quality); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCropInfo | Retrieves whether or not the cropping is enabled for the referenced  AEGP\_OutputModuleRefH, and the rectangle to be used.  AEGP\_GetCropInfo(  AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, A\_Boolean \*is\_enabledBP,  A\_Rect \*crop\_rectP); |
| AEGP\_SetCropInfo | Specifies whether cropping is enabled for the referenced  AEGP\_OutputModuleRefH, and the rectangle to be used.  AEGP\_SetCropInfo(  AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, A\_Boolean enableB,  A\_Rect crop\_rect); |
| AEGP\_GetSoundFormatInfo | Retrieves whether or not audio output is enabled for the referenced  AEGP\_OutputModuleRefH, and the settings to be used.  AEGP\_GetSoundFormatInfo( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, AEGP\_SoundDataFormat \*formatP,  A\_Boolean \*enabledPB); |
| AEGP\_SetSoundFormatInfo | Specifies whether or not audio output is enabled for the referenced  AEGP\_OutputModuleRefH, and the settings to be used.  AEGP\_SetSoundFormatInfo( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, AEGP\_SoundDataFormat format\_info, A\_Boolean enabledB); |
| AEGP\_GetOutputFilePath | Retrieves the path to which AEGP\_OutputModuleRefH‘s output file will be written. The path is a handle to a NULL-  terminated A\_UTF16Char string, and must be disposed with  AEGP\_FreeMemHandle.  AEGP\_GetOutputFilePath( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH,  AEGP\_MemHandle \*unicode\_pathPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetOutputFilePath | Specifies the path to which AEGP\_OutputModuleRefH‘s output file will be written. The file path is a NULL-terminated UTF-16  string with platform separators.  AEGP\_SetOutputFilePath( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, const A\_UTF16Char \*pathZ); |
| AEGP\_AddDefault OutputModule | Adds the default output module to the specified AEGP\_RQItemRefH, and returns the added output module’s AEGP\_OutputModuleRefH (you wouldn’t add it if you didn’t plan to mess around with it, would you?).  AEGP\_AddDefaultOutputModule( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH \*outmodPH); |
| AEGP\_GetExtraOutput ModuleInfo | Retrieves information about the output module. format\_uniPH  and info\_uniPH provide the textual description of, and  information about, the output module, formatted as the user would see it. format\_uniPH and info\_uniPH will contain NULL- terminated UTF16 strings, of which the caller must dispose.  AEGP\_GetExtraOutputModuleInfo( AEGP\_RQItemRefH rq\_itemH, AEGP\_OutputModuleRefH outmodH, AEGP\_MemHandle \*format\_uniPH,  AEGP\_MemHandle \*info\_uniPH,  A\_Boolean \*is\_sequenceBP,  A\_Boolean \*multi\_frameBP); |

**WORKING WITH EFFECTS**

These functions provide a way for effects (and AEGPs) to obtain information about the

context of an applied effect. NOTE: Any time you modify or rely on data from outside the normal render pipeline, you run the risk of dependency problems. There is no way for After

Effects to know that you depend on this external information; consequently, you will not be notified if it changes out from under you.

**TABLE 98: AEGP\_PFINTERFACESUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetEffectLayer | Obtain the layer handle of the layer to which the effect is applied.  AEGP\_GetEffectLayer(  PF\_ProgPtr effect\_ref,  AEGP\_LayerH \*layerPH); |
| AEGP\_GetNewEffectForEffect | Obtain the AEGP\_EffectRefH corresponding to the effect.  AEGP\_GetNewEffectForEffect( AEGP\_PluginID aegp\_plugin\_id, PF\_ProgPtr effect\_ref, AEGP\_EffectRefH \*effectPH); |
| AEGP\_ConvertEffectToCompTime | Retreive the composition time corresponding to the effect’s layer time.  AEGP\_ConvertEffectToCompTime( PF\_ProgPtr effect\_ref,  long what\_timeL, unsigned long time\_scaleLu, A\_Time \*comp\_timePT); |
| AEGP\_GetEffectCamera | Obtain the camera (if any) being used by After Effects to view the effect’s layer.  AEGP\_GetEffectCamera(  PF\_ProgPtr effect\_ref, const A\_Time \*comp\_timePT, AEGP\_LayerH camera\_layerPH); |
| AEGP\_GetEffectCameraMatrix | Obtain the transform used to move between the layer’s coordinate space and that of the containing composition.  AEGP\_GetEffectCameraMatrix( PF\_ProgPtr effect\_ref, const A\_Time \*comp\_timePT, A\_Matrix4 \*camera\_matrixP,  A\_FpLong \*dst\_to\_planePF,  A\_short \*plane\_widthPL,  A\_short \*plane\_heightPL);  NOTE: In cases where the effect’s input layer has square pixels, but is in a non-square pixel composition, you must correct for the pixel aspect ratio by premultiplying the  matrix by (1/parF, 1, 1). |

**AEGP\_GETEFFECTCAMERAMATRIX NOTES**

[The model view for the camera matrix is inverse of the matrix obtained from](#_bookmark711) [AEGP](#_bookmark711)\_GetEffectCameraMatrix(). Also note that our matrix is row-based; OpenGL’s is column-based.

**DO THIS MANY TIMES**

Utilizes multiple processors (if available) for your computations.

**TABLE 99: AEGP\_ITERATESUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNumThreads | Ask After Effects how many threads are currently available.  AEGP\_GetNumThreads(  A\_long \*num\_threadsPL); |
| AEGP\_IterateGeneric | Specify a function for After Effects to manage on multiple  processors. Can be any function pointer specified by fn\_func, taking the arguments listed below. See [*Private Data*](#_bookmark543) for a  description of how refconPV is used.  AEGP\_IterateGeneric(  A\_long iterationsL,  void \*refconPV,  A\_Err (\*fn\_func) (void \*refconPV,  A\_long thread\_indexL, A\_long i,  A\_long iterationsL)); |

**FILE IMPORT MANAGER SUITE**

The FIMSuite allows file types handled by AEGPs to appear as part of the After Effects

import dialog, and drag-and-drop messaging. These are not for use by AEIOs! Rather, they are for importing projects which are best represented as After Effects compositions.

**TABLE 100: AEGP\_FIMSUITE3**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RegisterImport Flavor | Registers the name of the file type(s) supported by the plug-in. Upon return, imp\_refP will be a valid opaque reference, or AE\_FIM\_ImportFlavorRef\_NONE.  AEGP\_RegisterImportFlavor(  const char \*nameZ, AE\_FIM\_ImportFlavorRef \*imp\_refP); |
| AEGP\_RegisterImport FlavorFileTypes | Registers an array of file types and file extensions (the two arrays need not be of equal length) supported by the AEGP.  AEGP\_RegisterImportFlavorFileTypes( AE\_FIM\_ImportFlavorRef imp\_ref,  long num\_filekindsL,  const AEIO\_FileKind \*kindsAP,  long num\_fileextsL,  const AEIO\_FileKind \*extsAP); |
| AEGP\_RegisterImport FlavorImportCallbacks | Register the AEGP functions which will respond to import of different filetypes.  AEGP\_RegisterImportFlavorImportCallbacks( AE\_FIM\_ImportFlavorRef ref,  AE\_FIM\_ImportFlags single\_flag, const AE\_FIM\_ImportCallbacks \*imp\_cbsP); |
| AEGP\_SetImportedItem | Designates an item as having been imported (possibly replacing an existing item), and sets associated import options.  AEGP\_SetImportedItem( AE\_FIM\_ImportOptions imp\_options,  AEGP\_ItemH imported\_itemH); |

**CHEATING: EFFECT USAGE OF AEGP SUITES**

As soon as we showed developers the initial implementation of AEGP suites, they wanted to “cheat” and use them from within effects. This is certainly possible, but please keep in mind that depending on factors outside the effect API (i.e., any information you get from the

AEGP APIs) can lead to trouble. If After Effects thinks an effect has all the information it needs to render, it won’t (for example) update its parameters based on changes made through an AEGP function. We’re actively working on this dependency issue for future versions, but bear it in mind as you write effects which “masquerade” as AEGPs.

Effects can use some AEGP suites to take advantage of camera and lighting information, as well as the [AEGP\_GetLayerParentComp](#_bookmark593) and [AEGP\_GetCompBGColor](#_bookmark577) functions. This should not be interpreted to mean that effects can use *any* AEGP suite calls. Also, see the

[Events chapter](#_bookmark509) for more information on effects adding keyframes.

[AEGP\_PFInterfaceSuite](#_bookmark710) is the starting point. The functions in this suite allow you to retrieve the AEGP\_LayerH for the layer to which the effect is applied, and the

AEGP\_EffectRefH for the instance of your effect. [AEGP\_RegisterWithAEGP](#_bookmark668) allows you to get an AEGP\_PluginID, which is needed for many AEGP calls.

**DEPENDING ON AEGP QUERIES**

One word: Don’t. Effects cannot allow the results of AEGP queries to control what is rendered, without appropriately storing those query results (usually in sequence data),

cancelling their own render, and forcing a re-render using the queried information. This is

tricky. Failure to do so will result in nasty, subtle caching bugs guaranteed to cause hair loss and weight gain.

**AEGP DETAILS**

**HAVE A COOKIE**

In cases where After Effects must preserve state information around the functions your

AEGP calls (as when an artisan is rendering a frame, or a keyframer is adding and removing a series of keyframes from the same stream), you’ll call begin() and end() functions.

Typically, the begin function will return an opaque identifier, or ‘cookie’, which you must

then pass to the functions being used. The end function will properly dispose of the cookie. See [AEGP\_StartAddKeyframes()](#_bookmark644) for an example.

**MODIFYING ITEMS IN THE RENDER QUEUE**

If you call [AEGP\_AddCompToRenderQueue](#_bookmark700) (from [AEGP\_RenderQueueSuite](#_bookmark699)), or if the

user manually adds or removes a composition from the render queue, all references to render queue items are invalidated. Similarly, adding or removing output modules invalidates any

such references for each render queue item.

**NAMES AND SOLIDS**

Solids have names in the After Effects UI, but not in their [PF\_LayerDef](#_bookmark228). Consequently, their names cannot be retrieved by [AEGP\_GetItemName](#_bookmark569) or [AEGP\_GetLayerName](#_bookmark594).

However, you can use the ItemH associated with them to [AEGP\_GetItemName](#_bookmark569).

**REPORTING ERRORS AND PROBLEMS**

Use AEGP\_ItemSuite>AEGP\_ReportInfo() to report information to users, and identify your plug-in. AEIO plug-ins use the msg\_func pointer contained in the AEIO\_BasicData they’re passed (with every function) instead.

**TRANSFORMS: WHAT HAPPENS FIRST?**

After Effects computes rotation based on auto-orientation (towards path, or point of interest), then computes Orientation, then computes X, Y, and Z rotation.

**ACCESSING PIXELS FROM EFFECT LAYER PARAMETERS**

Use [AEGP\_GetNewStreamValue](#_bookmark629) to get the layer’s layer\_id, then the new

[AEGP\_GetLayerFromLayerID](#_bookmark603) to get the AEGP\_LayerH.

8 : ARTISANS

*NOTE: If you’re considering developing an Artisan, please talk it over with us first.*

The Artisan API exposes function hooks necessary for a plug-in to provide rendered output of 3D layers, taking over completely from After Effects (which still handles all rendering of 2D layers). There can be only one Artisan per composition, chosen from within the

*Composition Settings* > *Advanced* dialog. Artisans render the 3D environment, asking After Effects for information about each element in the composition. As you might guess, this is a vast and tedious process. This API is not recommended for anyone without a strong need to override After Effects’ 3D rendering.

Artisans may share information with effects written to communicate with them, but effects may not initiate this communication. Many of the suites used by Artisans require a

rendering context which is generated only after all effects have been applied to the layer.

**INTERACTIVE ARTISANS**

These differ from standard artisans in that they handle all layers in a composition (not just those which the user has made 3D), and they will only ever be called for onscreen display, never for rendered final output (the rendering calls “fall through” to the default artisan).

### ARTISAN DATA TYPES

Below are the data types most commonly used in the Artisan API.

**TABLE 101: DATA TYPES USED IN THE ARTISAN API**

|  |  |
| --- | --- |
| **Type** | **Describes** |
| AEGP\_RenderLayerContextH | State information at the time of a render request, sent to an Artisan by After Effects. |
| PR\_RenderContextH | A collection of settings defining what is to be rendered, and how. |

**TABLE 101: DATA TYPES USED IN THE ARTISAN API**

|  |  |
| --- | --- |
| **Type** | **Describes** |
| AEGP\_SoundDataH | The audio settings used for a given layer. |
| AEGP\_RenderReceiptH AEGP\_FrameReceiptH | Used by Artisans when rendering. |
| AEGP\_WorldH | A frame of pixels. |
| AEGP\_RenderOptionsH | The settings associated with a render queue item. |

**HORZ? VERT?**

After Effects’ matrix is row based; OpenGL’s is column based. This means more work for you. Yay, billable hours!

**IMPLEMENTATION AND DESIGN**

An Artisan is nearly an application unto itself. Because we realized early in the After Effects

5.0 that there are many ways to approach the problems inherent in 3D rendering;

intersections and shading, for example. We provided an API with which we and third parties (yes, we really do use our own APIs) could implement any 3D rendering scheme desired.

**3D COMPOSITING, NOT MODELING**

After Effects is *not* a 3D modeling application. Users work in a responsive mode, switching

to higher quality only at for proofing or final output. Consider providing at least two quality modes, one for layout and another for final output. Be conscious of render time in low

quality mode.

**REGISTERING AN ARTISAN**

An Artisan is an AEGP, and has a single entry point. Artisans must also register their own function entry points and have a special callback for this purpose. See [AEGP\_RegisterArtisan()](#_bookmark560).

This tables shows the functions that Artisans can support as defined by

PR\_ArtisanEntryPoints: only [render\_func](#_bookmark737) is required.

**TABLE 102: ARTISAN ENTRY POINTS**

|  |  |
| --- | --- |
| **PR\_ArtisanEntryPoints** | |
| global\_setup\_func0 | Called only once, right after GP\_Main. The global data is common across all instances of the plug-in. If you allocate memory during Global Setup, you must free it during your global\_setdown\_func.  PR\_GlobalSetupFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_GlobalDataH \*global\_dataPH); |
| global\_setdown\_func0 | Dispose of any global data you allocated.  PR\_GlobalSetdownFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_GlobalDataH global\_dataH); |
| global\_do\_about\_func0 | Tell the world about yourself! Use in\_dataP>msg\_func to display your dialog.  PR\_GlobalDoAboutFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_GlobalDataH global\_dataH); |
| setup\_instance\_func0 | Allocate and instantiate any data specific to this instance of your Artisan.  PR\_InstanceSetupFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH, PR\_GlobalDataH global\_dataH,  PR\_InstanceFlags flags,  PR\_FlatHandle flat\_dataH0, PR\_InstanceDataH \*instance\_dataPH); |
| setdown\_instance\_func0 | Deallocate and free any data specific to this instance of your Artisan.  PR\_InstanceSetdownFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH, PR\_GlobalDataH global\_dataH, PR\_InstanceDataH instance\_dataH); |

|  |  |
| --- | --- |
| **PR\_ArtisanEntryPoints** | |
| flatten\_instance\_func0 | Flatten your data in preparation to being written to disk. (making sure it’s OS independent, if your Artisan is).  PR\_FlattenInstanceFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH, PR\_GlobalDataH global\_dataH, PR\_InstanceDataH instance\_dataH, PR\_FlatHandle \*flatH); |
| do\_instance\_dialog\_func0 | If your Artisan has a additional parameters (accessed through its Options dialog), this function will be called to get and set them.  PR\_DoInstanceDialogFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH, PR\_GlobalDataH global\_dataH, PR\_InstanceDataH instance\_dataH, PR\_DialogResult \*resultP);  PR\_DialogResultis either PR\_DialogResult\_NO\_CHANGE or PR\_DialogResult\_CHANGE\_MADE. |
| frame\_setup\_func0 | Perform any setup necessary to render a frame (called immediately before rendering).  PR\_FrameSetupFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH PR\_RenderContextH render\_contextH, PR\_GlobalDataH global\_dataH, PR\_InstanceDataH instance\_dataH, PR\_RenderDataH \*render\_dataPH); |
| frame\_setdown\_func0 | Dispose of any setup data allocated during frame\_setup (sent immediately after rendering).  PR\_FrameSetdownFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH PR\_RenderContextH render\_contextH, PR\_GlobalDataH global\_dataH, PR\_InstanceDataH instance\_dataH, PR\_RenderDataH render\_dataH); |

|  |  |
| --- | --- |
| **PR\_ArtisanEntryPoints** | |
| render\_func | Render the scene.  PR\_FrameRenderFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH PR\_RenderContextH render\_contextH, PR\_GlobalDataH global\_dataH, PR\_InstanceDataH instance\_dataH, PR\_RenderDataH render\_dataH); |
| query\_func0 | Artisans can draw their own projection axes, should the need arise. After Effects will call this function to obtain the transform between the composition world and those axes, as well as for a number of other functions related to on- and off-screen preview drawing (the former is relevant only to interactive artisans).  PR\_QueryFunc(  const PR\_InData \*in\_dataP, PR\_GlobalContextH global\_contextH, PR\_InstanceContextH instance\_contextH PR\_QueryContextH query\_contextH, PR\_QueryType query\_type,  PR\_GlobalDataH global\_dataH, PR\_InstanceDataH instance\_dataH);  PR\_QueryType can be one of the following: PR\_QueryType\_NONE = 0, PR\_QueryType\_TRANSFORM, PR\_QueryType\_INTERACTIVE\_WINDOW\_DISPOSE, PR\_QueryType\_INTERACTIVE\_WINDOW\_CLEAR, PR\_QueryType\_INTERACTIVE\_WINDOW\_FROZEN\_PROXY, PR\_QueryType\_INTERACTIVE\_SWAP\_BUFFER,  PR\_QueryType\_INTERACTIVE\_DRAW\_PROCS, PR\_QueryType\_PREPARE\_FOR\_LINE\_DRAWING, PR\_QueryType\_UNPREPARE\_FOR\_LINE\_DRAWING, PR\_QueryType\_GET\_CURRENT\_CONTEXT\_SAFE\_FOR\_LINE\_DRAWING,  PR\_QueryType\_GET\_ARTISAN\_QUALITY  New in CS6. |

**THE WORLD IS YOUR CANVAS**

[AEGP\_RenderTexture()](#_bookmark743) supplies the raw pixels of a layer, untransformed, into an

arbitrarily-sized buffer. AEGP\_RenderLayer() invokes the entire After Effects render pipeline, including transforms, masking, et cetera, providing the layer as it appears in its composition, in a composition-sized buffer. If the layer being rendered is 3D, the default

(Standard 3D) Artisan is invoked to perform any 3D geometrics. Your Artisan can use this

to render track matte layers, and apply them only in a strictly 2D sense, to the transformed 3D layer.

Before rendering, the Artisans that ship with After Effects apply an inverse transform to get square pixels, then re-apply the transform before display. For example, if the pixel aspect

ratio is 10/11 (DV NTSC), we multiply by 11/10 to get square pixels. We process and composite 3D layers, then re-divide to get back to the original pixel aspect ratio.

The following suite supplies the layers, compositions, texture and destination buffers. This is a vital suite for all artisans.

**TABLE 103: AEGP\_CANVASSUITE8**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCompToRender | Given the render context provided to the Artisan at render time, returns a handle to the composition.  AEGP\_GetCompToRender( PR\_RenderContextH render\_contextH, AEGP\_CompH \*compPH) |
| AEGP\_GetNumLayersToRender | Given the render context, returns the number of layers the Artisan needs to render.  AEGP\_GetNumLayersToRender( PR\_RenderContextH render\_contextH, A\_long \*num\_to\_renderPL) |
| AEGP\_GetNthLayerContextTo Render | Used to build a list of layers to render after determining the total number of layers that need rendering by the Artisan.  AEGP\_GetNthLayerContextToRender( PR\_RenderContextH render\_contextH, A\_long n,  AEGP\_RenderLayerContextH\*layer\_indexPH) |
| AEGP\_GetLayerFromLayer Context | Given a AEGP\_RenderLayerContextH,retrieves the  associated AEGP\_LayerH (required by many suite functions).  AEGP\_GetLayerFromLayerContext(  const PR\_RenderContextH render\_contextH, AEGP\_RenderLayerContextH layer\_contextH, AEGP\_LayerH \*layerPH); |
| AEGP\_GetLayerAndSubLayer FromLayerContext | Allows for rendering of sub-layers (as within a Photoshop file).  AEGP\_GetLayerAndSubLayerFromLayerContext( const PR\_RenderContextH render\_contextH, AEGP\_RenderLayerContextHlayer\_contextH, AEGP\_LayerH \*layerPH, AEGP\_SubLayerIndex \*sublayerP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetTopLayerFromLayer Context | [With collapsed geometrics “on” this gives the layer in the root com- position containing the layer context. With collapsed geometrics off](#_bookmark742) [this is the same as](#_bookmark742) AEGP\_GetLayerFromLayerContext.  AEGP\_GetTopLayerFromLayerContext(  const PR\_RenderContextH r\_contextH, AEGP\_RenderLayerContextH l\_contextH, AEGP\_LayerH \*layerPH); |
| AEGP\_GetCompRenderTime | Given the render context, returns the current point in (composition) time to render.  AEGP\_GetNthLayerIndexToRender( PR\_RenderContextH render\_contextH, A\_long \*time,  A\_long \*time\_step) |
| AEGP\_GetCompDestination Buffer | Given the render context, returns a buffer in which to place the final rendered output.  AEGP\_GetCompToRender(  PR\_RenderContextH render\_contextH, AEGP\_CompH compH,  PF\_EffectWorld \*dst); |
| AEGP\_GetROI | Given the render context provided to the Artisan at render time, returns a handle to the composition.  AEGP\_GetROI(  PR\_RenderContextH render\_contextH, A\_LegacyRect \*roiPR); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RenderTexture | Given the render context and layer, returns the layer texture. All parameters with a trailing ‘0’ are optional; the returned PF\_EffectWorld can be NULL.  AEGP\_RenderTexture(  PR\_RenderContextH render\_contextH, AEGP\_LayerH layerH, AEGP\_RenderHints render\_hints, A\_FloatPoint \*suggested\_scaleP0, A\_FloatRect \*suggsted\_src\_rectP0,  A\_Matrix3 \*src\_matrixP0, PF\_EffectWorld \*render\_bufferP);  AEGP\_RenderHints contains one or more of the following:  AEGP\_RenderHints\_NONE AEGP\_RenderHints\_IGNORE\_EXTENTS AEGP\_RenderHints\_NO\_TRANSFER\_MODE  AEGP\_RenderHints\_NO\_TRANSFER\_MODE prevents application of opacity & transfer mode; for use with RenderLayer calls. |
| AEGP\_DisposeTexture | Disposes of an acquired layer texture.  AEGP\_DisposeTexture(  PR\_RenderContextH render\_contextH, AEGP\_LayerH layerH,  AEGP\_WorldH \*dst0); |
| AEGP\_GetFieldRender | Returns the field settings of the given PR\_RenderContextH.  AEGP\_GetFieldRender(  PR\_RenderContextH render\_contextH, PF\_Field \*field); |
| AEGP\_ReportArtisan Progress | Given the render context provided to the Artisan at render time, returns a handle to the composition. Note: this is NOT thread-safe on Mac OS; only use this function when the current thread ID is 0.  AEGP\_ReportArtisanProgress( PR\_RenderContextH render\_contextH, A\_long countL,  A\_long totalL); |
| AEGP\_GetRenderDownsample Factor | Returns the downsample factor of the PR\_RenderContextH.  AEGP\_GetRenderDownsampleFactor( PR\_RenderContextH render\_contextH, AEGP\_DownsampleFactor \*dsfP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_IsBlankCanvas | Determines whether the PR\_RenderContextH is blank (empty).  AEGP\_IsBlankCanvas(  PR\_RenderContextH render\_contextH, A\_Boolean \*is\_blankPB); |
| AEGP\_GetRenderLayerTo WorldXform | Given a render context and a layer (at a given time), retrieves the 4 by 4 transform to move between their coordinate spaces.  AEGP\_GetRenderLayerToWorldXform( PR\_RenderContextH render\_contextH, AEGP\_RenderLayerContextH layer\_contextH, const A\_Time \*comp\_timeP,  A\_Matrix4 \*transform); |
| AEGP\_GetRenderLayerBounds | Retrieves the bounding rectangle of the layer\_contextH (at a given time) within the render\_contextH.  AEGP\_GetRenderLayerBounds( PR\_RenderContextH render\_contextH, AEGP\_RenderLayerContextH layer\_contextH, const A\_Time \*comp\_timeP,  A\_LegacyRect \*boundsP); |
| AEGP\_GetRenderOpacity | Returns the opacity of the given layer context at the given time, within the render context.  AEGP\_GetRenderOpacity( PR\_RenderContextH render\_contextH,  AEGP\_RenderLayerContextH layer\_contextH, const A\_Time \*comp\_timePT,  A\_FpLong \*opacityPF); |
| AEGP\_IsRenderLayerActive | Returns whether or not a given layer context is active within the render context, at the given time.  AEGP\_IsRenderLayerActive( PR\_RenderContextH render\_contextH, AEGP\_RenderLayerContextH layer\_contextH, const A\_Time \*comp\_timePT,  A\_Boolean \*activePB); |
| AEGP\_SetArtisan LayerProgress | Sets the progress information for a rendering Artisan. countL is the number of layers completed, num\_layersL is the total number of layers the Artisan is rendering.  AEGP\_SetArtisanLayerProgress( PR\_RenderContextH render\_contextH, A\_long countL,  A\_long num\_layersL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RenderLayerPlus | Similar to AEGP\_RenderLayer, but takes into account the  AEGP\_RenderLayerContextH.  AEGP\_RenderLayerPlus( PR\_RenderContextH r\_contextH, AEGP\_LayerH layerH, AEGP\_RenderLayerContextH l\_contextH, AEGP\_RenderHints render\_hints, AEGP\_WorldH \*bufferP); |
| AEGP\_GetTrackMatteContext | Retrieves the AEGP\_RenderLayerContextH for the specified render and fill contexts.  AEGP\_GetTrackMatteContext( PR\_RenderContextH rnder\_contextH, AEGP\_RenderLayerContextH fill\_contextH, AEGP\_RenderLayerContextH \*mattePH); |
| AEGP\_RenderTextureWith Receipt | Renders a texture into an AEGP\_WorldH, and provides an AEGP\_RenderReceiptH for the operation. The returned receiptPH must be disposed of with [AEGP\_DisposeRenderReceipt](#_bookmark745).  AEGP\_RenderTextureWithReceipt( PR\_RenderContextH render\_contextH, AEGP\_RenderLayerContextH layer\_contextH, AEGP\_RenderHints render\_hints, A\_FloatPoint \*suggested\_scaleP0,  A\_FloatRect \*suggest\_src\_rectP0,  A\_Matrix3 \*src\_matrixP0, AEGP\_RenderReceiptH \*receiptPH, AEGP\_WorldH \*dstPH); |
| AEGP\_GetNumberOfSoftware Effects | Returns the number of software effects applied in the given  AEGP\_RenderLayerContextH.  AEGP\_GetNumberOfSoftwareEffects( PR\_RenderContextH ren\_contextH, AEGP\_RenderLayerContextH lyr\_contextH, A\_short \*num\_sft\_FXPS); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_RenderLayerPlus WithReceipt | An improvement over AEGP\_RenderLayerPlus, this function also provides an AEGP\_RenderReceiptH for caching purposes.  AEGP\_RenderLayerPlusWithReceipt( PR\_RenderContextH render\_contextH, AEGP\_LayerH layerH, AEGP\_RenderLayerContextH layer\_contextH, AEGP\_RenderHints render\_hints, AEGP\_NumEffectsToRenderType num\_effectsS, AEGP\_RenderReceiptH \*receiptPH, AEGP\_WorldH \*bufferPH); |
| AEGP\_DisposeRenderReceipt | Frees an AEGP\_RenderReceiptH.  AEGP\_DisposeRenderReceipt( AEGP\_RenderReceiptH receiptH); |
| AEGP\_CheckRenderReceipt | Checks with After Effects’ internal caching to determine whether a given AEGP\_RenderReceiptH is still valid.  AEGP\_CheckRenderReceipt(  PR\_RenderContextH current\_contextH,  AEGP\_RenderLayerContextH  current\_lyr\_ctxtH, AEGP\_RenderReceiptH old\_receiptH, A\_Boolean check\_aceB, AEGP\_NumEffectsToRenderType num\_effectsS, AEGP\_RenderReceiptStatus  \*receipt\_statusP); |
| AEGP\_GenerateRender Receipt | Generates a AEGP\_RenderReceiptH for a layer as if the first  num\_effectsS have been rendered  AEGP\_GenerateRenderReceipt( PR\_RenderContextH current\_contextH,  AEGP\_RenderLayerContextH  current\_lyr\_contextH,  AEGP\_NumEffectsToRenderType  num\_effectsS, AEGP\_RenderReceiptH \*render\_receiptPH); |
| AEGP\_GetNumBinsToRender | Returns the number of bins After Effects wants the artisan to render.  AEGP\_GetNumBinsToRender(  const PR\_RenderContextH contextH, A\_long \*num\_binsPL); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetNthBin | Sets the given render context to be the n-th bin to be rendered by After Effects.  AEGP\_SetNthBin(  const PR\_RenderContextH contextH, A\_long n); |
| AEGP\_GetBinType | Retrieves the type of the given bin.  AEGP\_GetBinType(  const PR\_RenderContextH contextH, AEGP\_BinType \*bin\_typeP);  AEGP\_BinType will be one of the following:  AEGP\_BinType\_NONE AEGP\_BinType\_2D AEGP\_BinType\_3D |
| AEGP\_GetRenderLayer ToWorldXform2D3D | Retrieves the transform to correctly orient the layer being rendered with the output world. Pass TRUE for only\_2dB to constrain the transform to two dimensions.  AEGP\_GetRenderLayerToWorldXform2D3D( PR\_RenderContextH render\_contextH, AEGP\_RenderLayerContextHlayer\_contextH, const A\_Time \*comp\_timeP,  A\_Boolean only\_2dB,  A\_Matrix4 \*transformP); |
| *Functions below are for interactive artisans only* | |
| AEGP\_GetPlatformWindowRef | Retrieves the platform-specific window context into which to draw the given PR\_RenderContextH.  AEGP\_GetPlatformWindowRef(  const PR\_RenderContextH contextH, AEGP\_PlatformWindowRef \*window\_refP); |
| AEGP\_GetViewportScale | Retrieves the source-to-frame downsample factor for the given  PR\_RenderContextH.  AEGP\_GetViewportScale(  const PR\_RenderContextH contextH, A\_FpLong \*scale\_xPF,  A\_FpLong \*scale\_yPF); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetViewportOrigin | Retrieves to origin of the source, within the frame (necessary to  translate between the two), for the given PR\_RenderContextH.  AEGP\_GetViewportOrigin(  const PR\_RenderContextH contextH, A\_long \*origin\_xPL,  A\_long \*origin\_yPL); |
| AEGP\_GetViewportRect | Retrieves the bounding rectangle for the area to be drawn, for the given PR\_RenderContextH.  AEGP\_GetViewportRect(  const PR\_RenderContextH contextH, A\_LegacyRect \*v\_rectPR); |
| AEGP\_GetFallowColor | Retrieves the color used for the fallow regions in the given  PR\_RenderContextH.  AEGP\_GetFallowColor(  const PR\_RenderContextH contextH, PF\_Pixel8 \*fallow\_colorP); |
| AEGP\_GetInteractive Checkerboard | Retrieves whether or not the checkerboard is currently active for the given PR\_RenderContextH.  AEGP\_GetInteractiveCheckerboard( const PR\_RenderContextH contextH,  A\_Boolean \*cboard\_onPB); |
| AEGP\_GetInteractive Checkerboard Colors | Retrieves the colors used in the checkerboard.  AEGP\_GetInteractiveCheckerboardColors( const PR\_RenderContextH contextH, PF\_Pixel \*color1P,  PF\_Pixel \*color2P); |
| AEGP\_GetInteractive CheckerboardSize | Retrieves the width and height of one checkerboard square.  AEGP\_GetInteractiveCheckerboardSize( const PR\_RenderContextH contextH,  A\_u\_long \*cbd\_widthPLu,  A\_u\_long \*cbd\_heightPLu); |
| AEGP\_GetInteractiveCached Buffer | Retrieves the cached AEGP\_WorldH last used for the  PR\_RenderContextH.  AEGP\_GetInteractiveCachedBuffer( const PR\_RenderContextH contextH, AEGP\_WorldH \*buffer); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_ArtisanMustRender AsLayer | Determines whether or not the artisan must render the current  AEGP\_RenderLayerContextH as a layer.  AEGP\_ArtisanMustRenderAsLayer(  const PR\_RenderContextH contextH, AEGP\_RenderLayerContextH layer\_contextH, A\_Boolean \*use\_txturePB); |
| AEGP\_GetInteractive DisplayChannel | Returns which channels should be displayed by the interactive artisan.  AEGP\_GetInteractiveDisplayChannel( const PR\_RenderContextH contextH, AEGP\_DisplayChannelType \*channelP);  AEGP\_DisplayChannelType will be one of the following:  AEGP\_DisplayChannel\_NONE AEGP\_DisplayChannel\_RED AEGP\_DisplayChannel\_GREEN AEGP\_DisplayChannel\_BLUE AEGP\_DisplayChannel\_ALPHA AEGP\_DisplayChannel\_RED\_ALT AEGP\_DisplayChannel\_GREEN\_ALT AEGP\_DisplayChannel\_BLUE\_ALT AEGP\_DisplayChannel\_ALPHA\_ALT |
| AEGP\_GetInteractive Exposure | Returns the exposure for the given PR\_RenderContextH, expressed as a floating point number.  AEGP\_GetInteractiveExposure( const PR\_RenderContextH rcH,  A\_FpLong \*exposurePF); |
| AEGP\_GetColorTransform | Returns the color transform for the given PR\_RenderContextH. AEGP\_GetColorTransform)(  const PR\_RenderContextH render\_contextH, A\_Boolean \*cms\_onB,  A\_u\_long \*xform\_keyLu,  void \*xformP); |
| AEGP\_GetCompShutterTime | Returns the shutter angle for the given PR\_RenderContextH. AEGP\_GetCompShutterTime)(  PR\_RenderContextH render\_contextH, A\_Time \*shutter\_time,  A\_Time \*shutter\_dur); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_MapCompToLayerTime | New in CC. Unlike [AEGP\_ConvertCompToLayerTime](#_bookmark601), this handles time remapping with collapsed or nested comps.  AEGP\_MapCompToLayerTime( PR\_RenderContextH render\_contextH,  AEGP\_RenderLayerContextH layer\_contextH, const A\_Time \*comp\_timePT,  A\_Time \*layer\_timePT); |

**CONVERT BETWEEN DIFFERENT CONTEXTS**

Convert between render and instance contexts, and manage global data specific to the artisan.

**TABLE 104: AEGP\_ARTISANUTILSUITE1**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetGlobalContext FromInstanceContext | Given an instance context, returns a handle to the global context.  AEGP\_GetGlobalContextFromInstanceContext( const PR\_InstanceContextH  instance\_contextH,  PR\_GlobalContextH  \*global\_contextPH); |
| AEGP\_GetInstanceContext FromRenderContext | Given the render context, returns a handle to the instance context.  AEGP\_GetInstanceContextFromRenderContext( const PR\_RenderContextH render\_contextH, PR\_InstanceContextH \*instnc\_ctextPH); |
| AEGP\_GetInstanceContext FromQueryContext | Given a query context, returns a handle to the instance context.  AEGP\_GetInstanceContextFromQueryContext( const PR\_QueryContextH query\_contextH, PR\_InstanceContextH \*instnce\_contextPH); |
| AEGP\_GetGlobalData | Given the global context, returns a handle to global data.  AEGP\_GetGlobalData(  const PR\_GlobalContextH global\_contextH, PR\_GlobalDataH \*global\_dataPH); |
| AEGP\_GetInstanceData | Given an instance context, return the associated instance data.  AEGP\_GetInstanceData(  const PR\_InstanceContextH instance\_contextH, PR\_InstanceDataH \*instance\_dataPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetRenderData | Given a render context, returns the associated render data.  AEGP\_GetRenderData(  const PR\_RenderContextH render\_contextH, PR\_RenderDataH \*render\_dataPH); |

**SMILE! CAMERAS**

Obtains the camera geometry, including camera properties (type, lens, depth of field, focal distance, aperture, et cetera).

**TABLE 105: AEGP\_CAMERASUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetCamera | Given a layer handle and time, returns the current camera layer handle.  AEGP\_GetCamera(  PR\_RenderContextH render\_contextH, const A\_Time \*comp\_timeP,  AEGP\_LayerH \*camera\_layerPH); |
| AEGP\_GetCameraType | Given a layer, returns the camera type of the layer.  AEGP\_GetCameraType(  AEGP\_LayerH aegp\_layerH, AEGP\_CameraType \*camera\_typeP;  The camera type can be the following:  AEGP\_CameraType\_NONE = -1 AEGP\_CameraType\_PERSPECTIVE AEGP\_CameraType\_ORTHOGRAPHIC |
| AEGP\_GetDefaultCamera DistanceToImagePlane | Given a composition handle, returns the camera distance to the image plane.  AEGP\_GetDefaultCamera DistanceToImagePlane(  AEGP\_CompH compH,  A\_FpLong \*dist\_to\_planePF) |
| AEGP\_GetCameraFilmSize | Retrieves the size (and units used to measure that size) of the film used by the designated camera.  AEGP\_GetCameraFilmSize(  AEGP\_LayerH camera\_layerH, AEGP\_FilmSizeUnits \*film\_size\_unitsP, A\_FpLong \*film\_sizePF0); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetCameraFilmSize | Sets the size (and unites used to measure that size) of the film used by the designated camera.  AEGP\_SetCameraFilmSize)(  AEGP\_LayerH camera\_layerH, AEGP\_FilmSizeUnits film\_size\_units, A\_FpLong \*film\_sizePF0); |

**NOTES REGARDING CAMERA BEHAVIOR**

Camera orientation is in composition coordinates, and the rotations are in layer (the

camera’s layer) coordinates. If the camera layer has a parent, the position is in a coordinate space relative to the parent.

**ORTHOGRAPHIC CAMERA MATRIX**

Internally, we use composition width and height to set the matrix described by the OpenGL specification as

glOrtho(-width/2, width/2, -height/2, height/2, -1, 100);

The orthographic matrix describes the projection. The position of the camera is described by another, scaled matrix. The inverse of the camera position matrix provides the “eye”

coordinates.

**FOCUS ON FOCAL**

Remember, focal length affects field of view; focal distance only affects depth of field.

**FILM SIZE**

In the real world, film size is measured in millimeters. In After Effects, it’s measured in

pixels. Multiply by 72 and divide by 25.4 to move from millimeters to pixels. Field of view is more complex;

ϴ = 1/2 field of view

tan(ϴ) = 1/2 composition height / focal length focal length = 2 tan(ϴ) / composition height

**HIT THE LIGHTS!**

Get and set the type of lights in a composition.

**TABLE 106: AEGP\_LIGHTSUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetLightType | Retrieves the AEGP\_LightType of the specified camera layer.  AEGP\_GetLightType(  AEGP\_LayerH light\_layerH, AEGP\_LightType \*light\_typeP);  AEGP\_LightType will be one of the following:  AEGP\_LightType\_PARALLEL AEGP\_LightType\_SPOT AEGP\_LightType\_POINT AEGP\_LightType\_AMBIENT |
| AEGP\_SetLightType | Sets the AEGP\_LightType for the specified camera layer.  AEGP\_SetLightType(  AEGP\_LayerH light\_layerH, AEGP\_LightType light\_type); |

**NOTES ON LIGHT BEHAVIOR**

The formula for parallel lights is found in Foley and Van Dam’s “Introduction to Computer Graphics” (ISBN 0-201-60921-5) as is the formula for point lights. We use the half angle

variant proposed by Jim Blinn instead.

Suppose we have a point on a layer and want to shade it with the light. Let V be the unit vector from the layer point to the eye point. Let L be the unit vector to the light (in the

parallel light case this is constant). Let H be (V+L)/2 (normalized). Let N be the unit normal vector to the layer.The amount of specular reflected light is S \* power(H Dot N, shine), where S is the specular coefficient.

**HOW SHOULD I DRAW THAT?**

After Effects relies upon Artisans to draw 3D layer handles. If your Artisan chooses not to respond to this call, the default Artisan will draw 3D layer handles for you. Querying

transforms is important for optimization of After Effects’ caching.

The coordinate system is positive x to right, positive y down, positive z into the screen. The origin is the upper left corner. Rotations are x then y then z. For matrices the translate is the bottom row, orientations are quaternions (which are applied first), then any x-y-z rotation

after that. As a general rule, use orientation or rotation but not both. Also use rotations if you need control over angular velocity.

**QUERY TRANSFORM FUNCTIONS**

These functions give artisans information about the transforms they’ll need in order to

correctly place layers within a composition and respond appropriately to the various queries After Effects will send to their [PR\_QueryFunc](#_bookmark738) entry point function. As that entry point is optional, so is your artisan’s response to the queries; however, if you don’t, your users may be disappointed that (while doing interactive preview drawing) all the camera and light

indicators vanish, until they stop moving! Artisans are complex beasts; contact us if you have any questions.

**TABLE 107: AEGP\_QUERYXFORMSUITE2**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_QueryXformGetSrcType | Given a query context, returns trasnsform source currently being modified.  AEGP\_QueryXformGetSrcType( PR\_QueryContextH query\_contextH, AEGP\_QueryXformType \*src\_type);  The query context will be one of the following:  AEGP\_Query\_Xform\_LAYER, AEGP\_Query\_Xform\_WORLD, AEGP\_Query\_Xform\_VIEW, AEGP\_Query\_Xform\_SCREEN |
| AEGP\_QueryXformGetDstType | Given a query context, returns the currently requested transform destination.  AEGP\_QueryXformGetDstType( PR\_QueryContextH query\_contextH, AEGP\_QueryXformType \*dst\_type); |
| AEGP\_QueryXformGetLayer | Used if the source or destination type is a layer. Given a query context, returns the layer handle.  AEGP\_QueryXformGetLayer( PR\_QueryContextH query\_contextH, AEGP\_LayerH \*layerPH); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_QueryXformGetComp | Given a query context, returns the current composition handle.  AEGP\_QueryXformGetComp(  PR\_QueryContextH query\_contextH, AEGP\_CompH \*compPH); |
| AEGP\_QueryXformGet TransformTime | Given a query context, returns the time of the transformation.  AEGP\_QueryXformGetTransformTime( PR\_QueryContextH query\_contextH, A\_Time \*time); |
| AEGP\_QueryXformGet ViewTime | Given a query context, returns the time of the associated view.  AEGP\_QueryXformGetViewTime( PR\_QueryContextH query\_contextH, A\_Time \*time); |
| AEGP\_QueryXformGetCamera | Given a query context, returns the current camera layer handle.  AEGP\_QueryXformGetCamera( PR\_QueryContextH query\_contextH,  AEGP\_LayerH \*camera\_layerPH); |
| AEGP\_QueryXformGetXform | Given a query context, returns the current matrix transform.  AEGP\_QueryXformGetXform( PR\_QueryContextH query\_contextH, A\_Matrix4 \*xform); |
| AEGP\_QueryXformSetXform | Given a query context, return the matrix transform you compute in  xform.  AEGP\_QueryXformSetXform( PR\_QueryContextH query\_contextH, A\_Matrix4 \*xform); |
| AEGP\_QueryWindowRef | Sets the window reference to be used (by After Effects) for the given PR\_QueryContextH.  AEGP\_QueryWindowRef( PR\_QueryContextH q\_contextH,  AEGP\_PlatformWindowRef \*window\_refP); |
| AEGP\_QueryWindowClear | Returns which AEGP\_PlatformWindowRef (and A\_Rect) to clear, for the given PR\_QueryContextH.  AEGP\_QueryWindowClear( PR\_QueryContextH q\_contextH, AEGP\_PlatformWindowRef \*window\_refP, A\_LegacyRect \*boundsPR); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_QueryFrozenProxy | Returns whether or not the textures used in the given  PR\_QueryContextH should be frozen.  AEGP\_QueryFrozenProxy( PR\_QueryContextH q\_contextH, A\_Boolean \*onPB); |
| AEGP\_QuerySwapBuffer | Sent after rendering and camera/light handle drawing is complete; After Effects returns the buffer into which the artisan should draw its output.  AEGP\_QuerySwapBuffer(  PR\_QueryContextH q\_contextH, AEGP\_PlatformWindowRef \*window\_refP, AEGP\_WorldH \*dest\_bufferp); |
| AEGP\_QueryDrawProcs | Sets the interactive drawing functions After Effects will call while drawing camera and lighting handles into the artisan’s provided  context.  AEGP\_QueryDrawProcs(  PR\_QueryContextH query\_contextH, [PR\_InteractiveDrawProcs](#_bookmark762) \*window\_refP); |
| AEGP\_QueryPrepareFor LineDrawing | Informs After Effects about the context into which it will be drawing.  AEGP\_QueryPrepareForLineDrawing( PR\_QueryContextH query\_contextH, AEGP\_PlatformWindowRef \*window\_refP, A\_LegacyRect \*viewportP,  A\_LPoint \*originP,  A\_FloatPoint \*scaleP); |
| AEGP\_QueryUnprepareFor LineDrawing | As far as After Effects is concerned, the artisan is done drawing lines.  AEGP\_QueryUnprepareForLineDrawing( PR\_QueryContextH query\_contextH, AEGP\_PlatformWindowRef \*window\_refP); |

We’ve added the ability for artisans to provide functions After Effects can use to do basic

drawing functions for updating the comp window display during preview, including camera, light, and wireframe preview modeling.

**TABLE 108: PR\_INTERACTIVEDRAWPROCS**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| PR\_Draw\_MoveToFunc | PR\_Draw\_MoveToFunc( short x,  short y); |
| PR\_Draw\_LineToFunc | PR\_Draw\_LineToFunc( short x,  short y); |
| PR\_Draw\_ForeColorFunc | PR\_Draw\_ForeColorFunc(  const A\_Color \*fore\_color); |
| PR\_Draw\_FrameRectFunc | PR\_Draw\_FrameRectFunc(  const A\_Rect \*rectPR ); |
| PR\_Draw\_PaintRectFunc | PR\_Draw\_PaintRectFunc(  const A\_Rect \*rectPR ); |

**NOTES ON QUERY TIME FUNCTIONS**

AEGP\_QueryXformGetTransformTime() and AEGP\_QueryXformGetViewTime() are both necessary for an artisan to build a representation of the scene to render.

AEGP\_QueryXformGetTransformTime() gets the time of the transform, which is then

passed to [AEGP\_GetCompShutterFrameRange()](#_bookmark578). AEGP\_QueryXformGetViewTime() gets the time of the view, which is used in calling [AEGP\_GetLayerToWorldXformFromView()](#_bookmark602).

# 9 : AEIOS

AEIOs are AEGPs that perform media file import and/or export. AEIOs do everything for a file of a given type that After Effects (or the plug-ins which ship with After Effects) would

normally do. On the import side, AEIOs can open existing files, manage file-specific interpretation options, and provide audio and frames from the file to After Effects in

AEGP\_SoundWorld and PF\_EffectWorld format. Additionally, AEIOs can create files

interactively, asking users for the settings they’d like instead of reading them from a file. On the export side, AEIOs can create and manage output options for render queue items, create output files and save frames (provided by After Effects as PF\_EffectWorlds) into those

files.

AEIOs work with uncompressed video with pixels in ARGB order from low to high-byte.

Pixels can be 8-bit, 16-bit, or 32-bit float per channel. AEIOs must handle their own compression/decompression of any codecs supported.

**AEIO, OR AEGP?**

AEIOs provide pixels and audio data to After Effects. If you’re writing an importer/exporter for a file format that represents timeline or project format (referencing file formats

supported by After Effects or other installed AEIOs), write an AEGP and add its command to the Import/Export submenu.

**AEIO FOR IMPORT, OR MEDIACORE IMPORTER?**

After Effects supports MediaCore importer plug-ins. MediaCore is a set of shared libraries that grew out of Premiere Pro; thus the MediaCore APIs are described in the [Premiere Pro](http://www.adobe.com/devnet/premiere/)  [SDK](http://www.adobe.com/devnet/premiere/).

Only MediaCore importer plug-ins support an importer priority system: The highest priority importer gets the first opportunity to import a file, and if the particular imported file isn’t

supported, the next-highest priority importer will then have the opportunity to try importing it, and so on. MediaCore importers cannot defer file import to an AEIO. So if your goal is to take over file handling for any file type for which After Effects already provides a plug-in,

you need to develop a MediaCore importer plug-in.

On the other hand, only AEIOs can display a setup dialog in the Interpret Footage < Main

> More Options dialog.

If the above constraints haven’t already answered whether you need to build an AEIO or MediaCore importer, then you’ll likely want to build a MediaCore importer, which can be used across the video and audio applications including Premiere Pro, Media Encoder,

Prelude, SpeedGrade, and Audition.

**HOW IT WORKS**

From within its entry point function, an AEIO populates a structure of function pointers with the names of the functions it wants called in response to certain events. Many of these function hooks are optional.

**WHAT WOULD AFTER EFFECTS DO?**

For many AEIO hook functions, you can ask After Effects to perform default processing

(this capability is noted in each hook’s descriptions). Unless you have compelling reasons to do otherwise, return AEIO\_Err\_USE\_DFLT\_CALLBACK from the function, and let After

Effects do the work. This is also a good way to learn the calling sequence before beginning implementation.

**REGISTERING YOUR AEIO**

During your plug-in’s entry point function, populate a AEIO\_ModuleInfo describing the filetype(s) the AEIO supports, and an AEIO\_FunctionBlock structure that points to your file handling functions. For some of these functions, you can rely on After Effects’ default

behavior by returning AEIO\_Err\_USE\_DFLT\_CALLBACK. However, you must still provide a function matching the required signature, that does so. Once you’ve filled out both these

structures, call [AEGP\_RegisterIO()](#_bookmark561).

In the AEIO\_ModuleInfo that you pass in to the register call, you provide the file type and description information that After Effects uses in the Import dialog, for the “Files of type” drop-down on Windows, or the Enable drop-down on MacOS. As of CS6, file extensions

cannot be more than three characters long, even though we have a few built-in importers with longer extensions.

**INSPEC, OUTSPEC**

On most import-related functions, an AEIO\_InSpecH is passed. On most output-related

functions, an AEIO\_OutSpecH is passed. What are these mysterious handles? These opaque data handles can be used with the [AEGP\_IOInSuite](#_bookmark806) and [AEGP\_IOOutSuite](#_bookmark811), to set or

query for information about the import or output. For example, on an import, you’ll use

AEIO\_InSpecH when calling AEGP\_SetInSpecDimensions in AEGP\_IOInSuite. And

during an export, you’ll use AEIO\_OutSpecH when calling AEGP\_GetOutSpecDimensions

in AEGP\_IOOutSuite. So use these handles to exchange information with After Effects about the details of the input or output.

### CALLING SEQUENCE

As with all AEGPs, the entry point function exported in the plug-in’s PiPL is called during launch. During this function, the AEIO must provide function pointers to required

functions and describe their capabilities, then pass the appropriate structures to

[AEGP\_RegisterIO()](#_bookmark561).

**IMPORT**

When users select a file in the file import dialog which is of a type handled by your AEIO, its [AEIO\_VerifyFileImportable()](#_bookmark805) function will be called; it’s called again for each

such file the user imports. [AEIO\_InitInSpecFromFile()](#_bookmark785) will be called for each file; parse the file, and use the various set functions to describe it to After Effects. Also,

construct any options data associated with the file, and save that data using [AEGP\_SetInSpecOptionsHandle()](#_bookmark810). After Effects then calls the plug-in’s [AEIO\_GetInSpecInfo()](#_bookmark787) function, to get descriptive text about the file for display in the project window. As noted in the description of this function, it may be called for folders as well; we recommend that, if there is no valid options data for the file, you do nothing and

return no error (that’s what our AEIOs do).

[AEIO\_CountUserData()](#_bookmark803) is then sent; if the AEIO indicates that there is user data

present, [AEIO\_GetUserData()](#_bookmark804) will follow. After Effects will then request that the plug-in draw a frame of video (for the project window thumbnail) by sending [AEIO\_DrawSparseFrame()](#_bookmark788).

Once the supported file is added to a composition, user interaction will generate calls to

[AEIO\_DrawSparseFrame()](#_bookmark788) and [AEIO\_GetSound()](#_bookmark789).

When the project is saved, and if there is options data associated with the AEIO\_InSpec,

After Effects will send [AEIO\_FlattenOptions()](#_bookmark786) during which the AEIO parses the options data, and creates a representation of it that contains no references to external memory. Likewise, the presence of any AEIO\_OutSpec options data will result in [AEIO\_GetFlatOutputOptions()](#_bookmark791) being sent.

**EXPORT**

If the user adds an item to the render queue and chooses the AEIO’s supported output

format, [AEIO\_InitOutputSpec()](#_bookmark790) will be sent. Use the various get functions to obtain information about the output settings, and store any pertinent information using [AEGP\_SetOutSpecOptionsHandle()](#_bookmark813) , followed by [AEIO\_GetFlatOutputOptions()](#_bookmark791). [AEIO\_GetDepths()](#_bookmark802) is sent so After Effects can determine what output pixel bit depths the AEIO supports.

[AEIO\_GetOutputInfo()](#_bookmark793) is sent so that file name, type and subtype information can be displayed in the output module details.

When the user clicks on the Format Options button, in the render queue,

[AEIO\_UserOptionsDialog()](#_bookmark792) is called.

When the user actually clicks on the “Render” button, [AEIO\_SetOutputFile()](#_bookmark794) will be

called, followed by [AEIO\_GetSizes()](#_bookmark800) (your AEIO is responsible for determining whether the destination has sufficient disk space available).

Before the video frames are sent, [AEIO\_StartAdding()](#_bookmark795) is sent for the AEIO to open the file handle and write out the file header. If the AEIO supports a video or audio format,

[AEIO\_AddSoundChunk()](#_bookmark801) is sent for each audio chunk, and an [AEIO\_AddFrame()](#_bookmark796) for

each video frame. If the AEIO supports sequences of still images, [AEIO\_OutputFrame()](#_bookmark798) is called repeatedly. After Effects sends a PF\_EffectWorld representation of the frame to be output. [AEIO\_WriteLabels()](#_bookmark799) is called (for each frame) to give the plug-in a chance to write out field and alpha interpretation information. [AEIO\_EndAdding()](#_bookmark797) is sent when

there are no more frames (or audio) to be output. Close the output file.

**AEIO\_MODULEINFO**

This is the structure where your AEIO will define its basic properties. Notice that, in addition to describing the filetypes and extensions supported by your AEIO, you also describe your signature and behavior using the AEIO\_ModuleFlags. We love flags.

**TABLE 109: AEIO\_MODULEINFO**

|  |  |
| --- | --- |
| **Member** | **Purpose** |
| sig | A long, uniquely identifying your plug-in. Many developers prefer to use a decidedly Mac-ish four  character code here. Please [*let us know*](mailto:zlam@adobe.com) what sig you’re using. |
| name | Descriptive name for your AEIO plug-in. |
| flags | Set of [AEIO\_ModuleFlags](#_bookmark780). |
| flags2 | Set of [AEIO\_ModuleFlags2](#_bookmark782). |
| max\_width, max\_height | The maximum dimensions supported by your format. |
| num\_filetypes | The number of filetypes supported by your AEIO. |
| num\_extensions | The number of file extensions supported by your AEIO. |
| num\_clips | The number of clipboard formats supported by your AEIO. |
| create\_kind | The Mac OS four character code for files created by your AEIO. |
| create\_ext | The file extension for files created by your AEIO. |
| read\_kinds | This array of 16 AEIO\_FileKinds need not be  entirely filled out, but the first [num\_filetypes + num\_extensions + num\_clips] ones must be populated, in that order. |
| num\_aux\_extensions | The number of auxiliary extensions supported by your AEIO. Say, for example, that you’re writing an AEIO to import information from a 3D program that saves scene  information into a .123 file, and camera information into a .xyz file. The .xyz would be an auxiliary extension; it’s not necessary to get the rest of the scene information, but it’s associated with the .123 files. |
| aux\_ext | The file extension of the auxiliary filetype(s) supported by your AEIO. |

AEIOs set these flags (like effect plug-ins use global outflags) in AEIO\_ModuleInfo.flags

to indicate their behavior to After Effects. Some flags are only relevant to input, and some are only relevant to output.

**TABLE 110: AEIO\_MODULEFLAGS**

|  |  |  |
| --- | --- | --- |
| **Flag** | **Purpose** | **I or O?** |
| AEIO\_MFlag\_INPUT | AEIO is an input module. | Input! |
| AEIO\_MFlag\_OUTPUT | AEIO is an output module (one plug-in can be both). | Output! |
| AEIO\_MFlag\_FILE | Each clip imported directly corresponds to a file, somewhere. | Both |
| AEIO\_MFlag\_STILL | Supports still images, not video. | Output |
| AEIO\_MFlag\_VIDEO | Supports video images, not stills. | Output |
| AEIO\_MFlag\_AUDIO | Supports audio. | Output |
| AEIO\_MFlag\_NO\_TIME | Time information isn’t part of the file format. This would be the case with numbered stills, with individual frames imported based on the composition’s time settings. | Input |
| AEIO\_MFlag\_INTERACTIVE\_GET | A new input sequence necessitates user  interaction. This would be the case for a non- file-based input module. | Input |
| AEIO\_MFlag\_INTERACTIVE\_PUT | A new output sequence necessitates user  interaction. This would be the case for a non- file-based output module. | Output |
| AEIO\_MFlag\_CANT\_CLIP | The AEIO’s drawing functions cannot accept dimensions smaller than the requested  dimensions. | Input |
| AEIO\_MFlag\_MUST\_INTERACT\_PUT | The AEIO must display a dialog box, even if a valid options data handle is available. | Output |
| AEIO\_MFlag\_CANT\_SOUND\_ INTERLEAVE | The AEIO requires that all video data be processed, then sound data (instead of  interleaving the processing the video and audio). | Output |
| AEIO\_MFlag\_CAN\_ADD\_FRAMES\_ NON\_LINEAR | The AEIO supports adding non-sequential frames. | Output |
| AEIO\_MFlag\_HOST\_DEPTH\_DIALOG | The AEIO wants After Effects to display a bit- depth selection dialog. | Input |

|  |  |  |
| --- | --- | --- |
| **Flag** | **Purpose** | **I or O?** |
| AEIO\_MFlag\_HOST\_FRAME\_START\_ DIALOG | The AEIO wants After Effects to display a  dialog requesting that the user specify a starting frame. | Input |
| AEIO\_MFlag\_NO\_OPTIONS | The AEIO does not accept output options. | Output |
| AEIO\_MFlag\_NO\_PIXELS | The AEIO’s file format doesn’t actually store pixels. Currently unused as of CS6. | (unused) |
| AEIO\_MFlag\_ SEQUENCE\_OPTIONS\_OK | The AEIO will adopt the sequence options of its parent if a folder is selected. | Input |
| AEIO\_MFlag\_INPUT\_OPTIONS | The AEIO has user options associated with each input sequence. NOTE: the options information must be flat (not referring to any data contained in external pointers or handles). | Input |
| AEIO\_MFlag\_HSF\_AWARE | The AEIO will provide horizontal scaling factor (pixel aspect ratio) information for each new  sequence. This prevents After Effects from guessing. | Input |
| AEIO\_MFlag\_HAS\_LAYERS | The AEIO supports multiple layers in a single document. | Input |
| AEIO\_MFlag\_SCRAP | The AEIO has a clipboard parsing component. | Input |
| AEIO\_MFlag\_NO\_UI | After Effects should display no UI for this module (do not combine this flag with  AEIO\_MFlag\_HOST\_DEPTH\_DIALOG or AEIO\_MFlag\_HOST\_FRAME\_ START\_DIALOG) | Input |
| AEIO\_MFlag\_SEQ\_OPTIONS\_DLG | The AEIO has sequence options accessible from the More Options button in the Interpret Footage dialog. | Input |
| AEIO\_MFlag\_HAS\_AUX\_DATA | The file format supported by the AEIO has  depth information, normals, or some other non- color information related to each pixel. | Input |
| AEIO\_MFlag\_HAS\_META\_DATA | The file format supported by the AEIO  supports user-definable metadata. If this flag is set, the embed pop-up in the output module  dialog will be enabled. | Output |
| AEIO\_MFlag\_CAN\_DO\_MARKERS | The file format support by the AEIO supports markers, url flips, and/or chapters. | Output |

|  |  |  |
| --- | --- | --- |
| **Flag** | **Purpose** | **I or O?** |
| AEIO\_MFlag\_CAN\_DRAW\_DEEP | The AEIO can draw into 16bpc (“deep”)  PF\_EffectWorlds. | Input |
| AEIO\_MFlag\_RESERVED4 | Special super-secret flag. Doesn’t do anything...or does it?  (*No, it doesn’t.*) | ??? |

**AEIO\_MODULEFLAGS2**

Gotta have dem flags...

**TABLE 111: AEIO\_MODULEFLAGS2**

|  |  |  |
| --- | --- | --- |
| **Flag** | **Purpose** | **I or O?** |
| AEIO\_MFlag2\_AUDIO\_OPTIONS | The AEIO has an audio options dialog. | Output |
| AEIO\_MFlag2\_SEND\_ADDMARKER\_ BEFORE\_ADDFRAME | The AEIO wants to receive marker data before outputting video or audio (useful for MPEG  streams). | Output |
| AEIO\_MFlag2\_CAN\_DO\_MARKERS\_2 | The AEIO supports combined markers; URL flips, chapters, and comments. | Output |
| AEIO\_MFlag2\_CAN\_DRAW\_FLOAT | The AEIO can draw into float (32-bpc) worlds. | Input |
| AEIO\_MFlag2\_CAN\_DO\_AUDIO\_32 | Supports 32-bit audio output. | Output |
| AEIO\_MFlag2\_SUPPORTS\_ICC\_ PROFILES | Supports ICC profiles. | Both |
| AEIO\_MFlag2\_CAN\_DO\_MARKERS\_3 | The AEIO supports combined markers; URL flips, chapters, comments, and cue points. | Output |
| AEIO\_MFlag2\_SEND\_ADDMARKER\_ BEFORE\_STARTADDING | The AEIO wants to process markers before video during export. | Output |
| AEIO\_MFlag2\_USES\_QUICKTIME | On MacOS, prior to the host calling [AEIO\_AddFrame](#_bookmark796) or [AEIO\_OutputFrame](#_bookmark798), it will lock the global QuickTime mutex. | Output |

### NEW KIDS ON THE FUNCTION BLOCK

During its main entry point function, each AEIO plug-in must fill in an AEIO\_FunctionBlock, providing pointers to the functions After Effects will call for different file-related tasks.

The table below shows which functions are needed for input, and which ones are needed for output. For a bare-bones implementation, start with the functions that are noted as

“Required” in the right column. You can often invoke “best-case” behavior by having After Effects handle the call for you (by returning AEIO\_Err\_USE\_DFLT\_CALLBACK).

For a barebones AEIO for video input only, implement the following functions:

AEIO\_InitInSpecFromFile or AEIO\_InitInSpecInteractive (depending on whether the

source is a file or interactively generated), AEIO\_DisposeInSpec, AEIO\_GetInSpecInfo,

AEIO\_DrawSparseFrame, AEIO\_CloseSourceFiles, and AEIO\_InqNextFrameTime (using AEIO\_Err\_USE\_DFLT\_CALLBACK is fine).

Starting from the IO sample, it is best to leave the other functions defined too, and fill them in further as needed.

**TABLE 112: AEIO\_FUNCTIONBLOCK4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_InitInSpec- FromFile | Given a file path, describe its contents to After Effects in the provided AEIO\_InSpecH. Use all appropriate “set” calls from the [AEIO\_AEGPIOInSuite](#_bookmark806)) to do so; if there is  image data, set its depth, dimensions, and alpha  interpretation. If there is audio, describe its channels and sample rate.  The file path is a NULL-terminated UTF-16 string with platform separators.  AEIO\_InitInSpecFromFile( AEIO\_BasicData \*basic\_dataP, const A\_UTF16Char \*file\_pathZ, AEIO\_InSpecH inH); | Input | Yes, for  file-based media |
| AEIO\_InitInSpec- Interactive | Using some form of user interaction (and not a file path provided by After Effects), describe the audio and video your generated AEIO\_InSpecH  contains.  AEIO\_InitInSpecInteractive( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH); | Input | Yes, for  interactiv ely  generated media |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Dispose- InSpec | Free an AEIO\_InSpecH.  AEIO\_DisposeInSpec(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH); | Input | Yes |
| AEIO\_Flatten- Options | For the given AEIO\_InSpecH, return a flattened version of the data contained in its options handle. Obtain the unflattened options handle using [AEGP\_GetInSpecOptionsHandle](#_bookmark809).  AEIO\_FlattenOptions( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH, AEIO\_Handle \*flat\_optionsPH); | Input | No |
| AEIO\_Inflate- Options | For the given AEIO\_InSpecH, create (using [AEGP\_SetInSpecOptionsHandle](#_bookmark810)) an unflattened version of its flattened option data.  AEIO\_InflateOptions( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH, AEIO\_Handle flat\_optionsH); | Input | No |
| AEIO\_SynchInSpec | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Inspect the AEIO\_InSpecH, update its options if necessary), and indicate whether or not you made  changes.  AEIO\_SynchInSpec(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_Boolean \*changed0); | Input | No |
| AEIO\_Get- ActiveExtent | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Populate the provided A\_LRect with the active extent of the file’s pixels at the given time.  AEIO\_GetActiveExtent( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  const A\_Time \*tr,  A\_LRect \*extent); | Input | Yes |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Get- InSpecInfo | Provide a few strings in AEIO\_Verbiage to  describe the file, which will appear in the Project  panel. This includes the strings used to describe the file type and subtype (the codec).  AEIO\_GetInSpecInfo(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH, AEIO\_Verbiage \*verbiageP);  This function gets called OFTEN; every time we refresh the project panel. Keep allocations to a  minimum. In the AEIOs that ship with After Effects, we check for a valid optionsH (using  [AEGP\_GetInSpecOptionsHandle](#_bookmark809)); if we find one, we use the information from within it. If not, we do nothing. This is important; if your AEIO  handles still images, this function *will* get called for the folder containing the stills. Hopefully, there  won’t be an optionsH associated with it (unless you’re writing a truly bizarre AEIO). | Input | Yes |
| AEIO\_Draw- SparseFrame | Draw a frame from the AEIO\_InSpecH. The PF\_EffectWorld\* contains the width and height to use, but make sure you take the required\_region0 into account, if it’s not NULL.  AEIO\_DrawSparseFrame( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  AEIO\_Quality qual,  const AEIO\_RationalScale \*rs0, const A\_Time \*tr,  const A\_Time \*duration0, const A\_Rect  \*required\_region0, PF\_EffectWorld \*wP, A\_long\* originx,  A\_long\* originy, AEIO\_DrawingFlags  \*draw\_flagsP);  NOTE: return data as linear light (1.0), and After Effects will perform any necessary transformations to bring the footage into the working colorspace. | Input | Yes |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Get- Dimensions | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Provide the dimensions (and, if necessary, scaling factor) of the video in the AEIO\_InSpecH.  AEIO\_GetDimensions(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  const AEIO\_RationalScale \*rs0, A\_long \*width0,  A\_long \*height0); | Input | No |
| AEIO\_GetDuration | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Provide the duration of an AEIO\_InSpecH, in seconds.  AEIO\_GetDuration(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_Time \*trP); | Input | No |
| AEIO\_GetTime | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Provide the timebase of an AEIO\_InSpecH.  AEIO\_GetTime(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_Time \*tr);  Here are the values we use internally for common timebases:  29.97 fps: scale = 100; value= 2997;  59.94 fps: scale = 50; value = 2997;  23.976 fps: scale = 125; value = 2997;  30 fps: scale = 1; value = 30;  25 fps: scale = 1; value = 25; | Input | No |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_GetSound | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Provide sound from an AEIO\_InSpecH, at the quality described.  AEIO\_GetSound(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH, AEIO\_SndQuality quality, const AEIO\_InterruptFuncs  \*interrupt\_funcsP0, const A\_Time \*startPT, const A\_Time \*durPT,  A\_u\_long start\_sampLu,  A\_u\_long num\_samplesLu,  void \*dataPV);  AEIO\_SndQuality may be: AEIO\_SndQuality\_APPROX, (this quality is used to draw the audio waveform) AEIO\_SndQuality\_LO, AEIO\_SndQuality\_HI | Input | No |
| AEIO\_Inq- NextFrameTime | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Provide the time of the next frame (in the source footage’s timebase) within the AEIO\_InSpecH.  AEIO\_InqNextFrameTime( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  const A\_Time \*base\_time\_tr,  AEIO\_TimeDir time\_dir,  A\_Boolean \*found0,  A\_Time \*key\_time\_tr0); | Input | Yes |
| AEIO\_Init- OutputSpec | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Perform any initialization necessary for a new AEIO\_OutSpecH, and indicate whether you made changes.  AEIO\_InitOutputSpec( AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH, A\_Boolean \*user\_interacted);  NOTE: The first time your AEIO is used, After Effects caches the last-known-good optionsH in its preferences. When testing this function, [*delete*](#_bookmark43) [*your preferences*](#_bookmark43) often. | Output | Yes |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Get- FlatOutputOptions | Describe (in an AEIO\_Handle) the output options for an AEIO\_OutSpecH, in a disk-safe flat data  structure (one that does not reference external  memory). Note that your output options must be cross-platform, so pay attention to byte ordering issues.  AEIO\_GetFlatOutputOptions( AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  AEIO\_Handle \*optionsH); | Output | Yes |
| AEIO\_Dispose- OutputOptions | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Free the memory for the output options passed in.  AEIO\_DisposeOutputOptions( AEIO\_BasicData \*basic\_dataP, void \*optionsPV); | Output | No |
| AEIO\_UserOptions- Dialog | Display an output settings dialog (select TIFF  output within After Effects to see when this dialog will occur). Store this information in an options  handle using  [AEGP\_SetInSpecOptionsHandle](#_bookmark810).  AEIO\_UserOptionsDialog( AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH, PF\_EffectWorld \*sample0, A\_Boolean  \*interacted0); | Output | No |
| AEIO\_Get- OutputInfo | Describe (in text) the output options in an  AEIO\_OutSpecH.  AEIO\_GetOutputInfo(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH, AEIO\_Verbiage \*verbiage); | Output | Yes |
| AEIO\_OutputInfo- Changed | Update the AEIO\_OutSpecH based on the current settings (using the various Get functions to obtain  them).  AEIO\_OutputInfoChanged( AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH); | Output | No |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Set- OutputFile | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Set the file path for output of an AEIO\_OutSpecH.  Return AEIO\_Err\_USE\_DEFAULT\_CALLBACK  unless you’ve changed the path.  The file path is a NULL-terminated UTF-16 string with platform separators.  AEIO\_SetOutputFile(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_UTF16Char \*file\_pathZ); | Output | Yes |
| AEIO\_StartAdding | Prepare to add frames to the output file. This is a  good time to create the ouput file(s) on disk, and to write any header information to such files. This is also your first opportunity to allocate pixel buffers based on valid output spec values.  AEIO\_StartAdding(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_long flags); | Output | Yes, for writing formats that  support multiple frames |
| AEIO\_AddFrame | Add frame(s) to output file. You may pass a pointer to a function you want called if the user interrupts the render.  AEIO\_AddFrame(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_long frame\_index,  A\_long frames, PF\_EffectWorld \*wP, const A\_LPoint \*origin0,  A\_Boolean was\_compressedB, AEIO\_InterruptFuncs \*inter0); | Output | Yes, for writing formats that  support multiple frames |
| AEIO\_EndAdding | Perform any clean-up associated with adding frames.  AEIO\_EndAdding(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_long flags); | Output | Yes, for writing formats that  support multiple frames |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_OutputFrame | Output a single frame.  AEIO\_OutputFrame(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH, PF\_EffectWorld \*wP); | Output | Yes, for writing formats that  support a single  frame |
| AEIO\_WriteLabels | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Set alpha interpretation and field usage information for the AEIO\_OutSpecH. Indicate in AEIO\_LabelFlags which flags you wrote.  AEIO\_WriteLabels(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH, AEIO\_LabelFlags \*written); | Output | Yes |
| AEIO\_GetSizes | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed. Provide information about file size and remaining free space on output volume.  AEIO\_GetSizes(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_u\_longlong \*free\_space,  A\_u\_longlong \*file\_size); | Output | Yes |
| AEIO\_Flush | Destroy any options or user data associated with the  OutSpecH.  AEIO\_Flush(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH); | Output | Yes |
| AEIO\_Add- SoundChunk | Add the given sound to the output file.  AEIO\_AddSoundChunk(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  const A\_Time \*start, AEIO\_SndWorldH swH); | Output | Yes, for writing formats with  audio |
| AEIO\_Idle | Optional. Do something with idle time. AEIO\_Err\_USE\_DFLT\_CALLBACK is not supported.  AEIO\_Idle(  AEIO\_BasicData \*basic\_dataP, AEIO\_ModuleSignature sig, AEIO\_IdleFlags  \*idle\_flags0); | Output | No |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_GetDepths | Set AEIO\_OptionsFlags to indicate which pixel and color depths are valid for your output format. See the discussion on [*Export Bit-Depth*](#_bookmark815).  AEIO\_GetDepths(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH, AEIO\_OptionsFlags \*which); | Output | Yes |
| AEIO\_Get- OutputSuffix | AEIO\_Err\_USE\_DFLT\_CALLBACK allowed.  Describe the three character extension for the output file.  AEIO\_GetOutputSuffix( AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_char \*suffix); | Output | Yes |
| AEIO\_SeqOptions- Dlg | Display a footage options dialog, and indicate whether the user made any changes.  AEIO\_SeqOptionsDlg(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_Boolean \*interactedPB); | Input | No |
| AEIO\_GetNum- AuxChannels | Enumerate the auxiliary (beyond red, green, blue and alpha) channels of data contained in an AEIO\_InSpecH.  AEIO\_GetNumAuxChannels( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_long \*num\_channelsPL); | Input | No |
| AEIO\_Get- AuxChannelDesc | Describe the data type, name, channel, and dimensionality of an auxiliary data channel.  AEIO\_GetAuxChannelDesc( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  long chan\_indexL, PF\_ChannelDesc \*descP); | Input | No |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Draw- AuxChannel | Draw the auxiliary channel(s) from an  AEIO\_InSpecH.  AEIO\_DrawAuxChannel( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_long chan\_indexL, const AEIO\_DrawFramePB \*pbP, PF\_ChannelChunk \*chunkP); | Input | No |
| AEIO\_Free- AuxChannel | Free data associated with an auxiliary channel.  AEIO\_FreeAuxChannel( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH, PF\_ChannelChunk \*chunkP); | Input | No |
| AEIO\_NumAuxFiles | Enumerate the files needed to render the given AEIO\_InSpecH. This function and AEIO\_GetNthAuxFileSpec will be called when the user chooses File > Dependencies >  Collect Files… Here your AEIO tells AE what the associated files are.  AEIO\_NumAuxFiles(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH seqH,  A\_long \*files\_per\_framePL); | Input | No |
| AEIO\_GetNth- AuxFileSpec | Retrieve data from the nth auxiliary file, for the specified frame. The path is a handle to a NULL- terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEIO\_GetNthAuxFileSpec( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH seqH,  A\_long frame\_numL,  A\_long n,  AEGP\_MemHandle \*pathPH); | Input | No, if no aux files |
| AEIO\_Close- SourceFiles | Close (or open, depending upon closeB) the  source files for an AEIO\_InSpecH. When the user Collects Files, the AEIO will first be asked to close its source files, then re-open them.  AEIO\_CloseSourceFiles( AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH seqH,  A\_Boolean closeB);  TRUE for close, FALSE for open. | Input | Yes |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Count- UserData | Enumerate the units of user data associated with the  AEIO\_InSpecH.  AEIO\_CountUserData(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_u\_long typeLu,  A\_u\_long max\_sizeLu,  A\_u\_long \*num\_of\_typePLu); | Input | No |
| AEIO\_SetUserData | Set user data (of the given index and type) for the given AEIO\_OutSpecH.  AEIO\_SetUserData(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_u\_long typeLu,  A\_u\_long indexLu, const AEIO\_Handle dataH); | Output | No |
| AEIO\_GetUserData | Describe the user data (at the index and of the type given) associated with the AEIO\_InSpecH.  AEIO\_GetUserData(  AEIO\_BasicData \*basic\_dataP, AEIO\_InSpecH inH,  A\_u\_long typeLu,  A\_u\_long indexLu,  A\_u\_long max\_sizeLu,  AEIO\_Handle \*dataPH); | Input | No |
| AEIO\_AddMarker | Associate a marker of the specified type, at the  specified frame, with the AEIO\_OutSpecH. You may provide an interrupt function to handle user cancellation of this action.  AEIO\_AddMarker(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_long frame\_index, AEIO\_MarkerType marker\_type, void \*marker\_dataPV, AEIO\_InterruptFuncs  \*inter0); | Output | No |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Response** | **I or O?** | **Required?** |
| AEIO\_Verify- FileImportable | Indicate (by setting importablePB) whether  or not the plug-in can import the file. Note that After Effects has already done basic extension checking; you may wish to open the file and  determine whether or not it’s valid. This can be a time-consuming process; most AEIOs that ship with After Effects simply return TRUE, and deal with bad files during [AEIO\_InitInSpecFromFile](#_bookmark785).  The file path is a NULL-terminated UTF-16 string with platform separators.  AEIO\_VerifyFileImportable( AEIO\_BasicData \*basic\_dataP,  AEIO\_ModuleSignature  sig, const A\_UTF16Char  \*file\_pathZ,  A\_Boolean \*importablePB); | Input | No |
| AEIO\_UserAudio- OptionsDialog | Display an audio options dialog.  AEIO\_UserAudioOptionsDialog( AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH, A\_Boolean \*interacted0); | Output | No |
| AEIO\_AddMarker3 | Add a marker, with a flag specifying whether or not this is a composition marker.  AEIO\_AddMarker3(  AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_long frame\_index, AEGP\_ConstMarkerValP marker\_valP, AEIO\_RenderMarkerFlag marker\_flag, AEIO\_InterruptFuncs \*inter0); | Output | No |
| AEIO\_GetMimeType | Describe the output mime type. This is used for XMP support.  AEIO\_GetMimeType( AEIO\_BasicData \*basic\_dataP, AEIO\_OutSpecH outH,  A\_long mime\_type\_sizeL,  char \*mime\_typeZ); | Output | No |

These functions manage an input specification, After Effects’ internal representation of data gathered from any source. Any image or audio data in After Effects (except solids) is

obtained from an input specification handle, or AEIO\_InSpecH

**TABLE 113: AEGP\_IOINSUITE5**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetInSpecOptionsHandle | Retrieves the options data (created by your AEIO) for the given AEIO\_InSpecH.  AEGP\_GetInSpecOptionsHandle( AEIO\_InSpecH inH,  void \*\*optionsPPV); |
| AEGP\_SetInSpecOptionsHandle | Sets the options data for the given AEIO\_InSpecH. Must be allocated using the [*MemorySuite*](#_bookmark549).  AEGP\_SetInSpecOptionsHandle( AEIO\_InSpecH inH,  void \*optionsPV,  void \*\*old\_optionsPPV); |
| AEGP\_GetInSpecFilePath | Retrieves the file path for the AEIO\_InSpecH. The file path is a handle to a NULL-terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  AEGP\_GetInSpecFilePath( AEIO\_InSpecH inH, AEGP\_MemHandle \*file\_nameZ); |
| AEGP\_GetInSpecNativeFPS | Retrieves the frame rate of the AEIO\_InSpecH.  AEGP\_GetInSpecNativeFPS( AEIO\_InSpecH inH,  A\_Fixed \*native\_fpsP); |
| AEGP\_SetInSpecNativeFPS | Sets the frame rate of the AEIO\_InSpecH.  AEGP\_SetInSpecNativeFPS( AEIO\_InSpecH inH,  A\_Fixed native\_fpsP); |
| AEGP\_GetInSpecDepth | Retrieves the bit depth of the image data in the  AEIO\_InSpecH.  AEGP\_GetInSpecDepth( AEIO\_InSpecH inH,  A\_short \*depthPS); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetInSpecDepth | Indicates to After Effects the bit depth of the image data in the AEIO\_InSpecH.  AEGP\_SetInSpecDepth( AEIO\_InSpecH inH,  A\_short depthS); |
| AEGP\_GetInSpecSize | Retrieves the size (in bytes) of the data referenced by the  AEIO\_InSpecH.  AEGP\_GetInSpecSize( AEIO\_InSpecH inH, AEIO\_FileSize \*sizePLLu); |
| AEGP\_SetInSpecSize | Indicates to After Effects the size (in bytes) of the data referenced by the AEIO\_InSpecH.  AEGP\_SetInSpecSize( AEIO\_InSpecH inH,  AEIO\_FileSize sizeL); |
| AEGP\_GetInSpecInterlaceLabel | Retrieves field information for the AEIO\_InSpecH.  AEGP\_GetInSpecInterlaceLabel( AEIO\_InSpecH inH,  FIEL\_Label \*interlaceP); |
| AEGP\_SetInSpecInterlaceLabel | Specifies field information for the AEIO\_InSpecH.  AEGP\_SetInSpecInterlaceLabel( AEIO\_InSpecH inH,  const FIEL\_Label \*interlaceP); |
| AEGP\_GetInSpecAlphaLabel | Retrieves alpha channel interpretation information for the  AEIO\_InSpecH.  AEGP\_GetInSpecAlphaLabel( AEIO\_InSpecH inH, AEIO\_AlphaLabel \*alphaP); |
| AEGP\_SetInSpecAlphaLabel | Sets alpha channel interpretation information for the  AEIO\_InSpecH.  AEGP\_SetInSpecAlphaLabel( AEIO\_InSpecH inH,  const AEIO\_AlphaLabel\*alphaP); |
| AEGP\_GetInSpecDuration | Retrieves the duration of the AEIO\_InSpecH.  AEGP\_GetInSpecDuration( AEIO\_InSpecH inH,  A\_Time \*durationP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetInSpecDuration | Sets the duration of the AEIO\_InSpecH. NOTE: As of  5.5, this must be called, even for frame-based file formats. If you don’t set the A\_Time.scale to something other than zero, your file(s) will not import. This will be fixed in  future versions.  AEGP\_SetInSpecDuration( AEIO\_InSpecH inH,  const A\_Time \*durationP); |
| AEGP\_GetInSpecDimensions | Retrieves the width and height of the image data in the  AEIO\_InSpecH.  AEGP\_GetInSpecDimensions( AEIO\_InSpecH inH,  A\_long \*widthPL0,  A\_long \*heightPL0); |
| AEGP\_SetInSpecDimensions | Indicates to After Effects the width and height of the image data in the AEIO\_InSpecH.  AEGP\_SetInSpecDimensions( AEIO\_InSpecH inH,  A\_long widthL,  A\_long heightL); |
| AEGP\_InSpecGetRational Dimensions | Retrieves the width, height, bounding rect, and scaling factor applied to an AEIO\_InSpecH.  AEGP\_InSpecGetRationalDimensions( AEIO\_InSpecH inH, const AEIO\_RationalScale \*rs0, A\_long \*width0,  A\_long \*height0,  A\_Rect \*r0); |
| AEGP\_GetInSpecHSF | Retrieves the horizontal scaling factor applied to an  AEIO\_InSpecH.  AEGP\_GetInSpecHSF( AEIO\_InSpecH inH,  A\_Ratio \*hsf); |
| AEGP\_SetInSpecHSF | Sets the horizontal scaling factor of an AEIO\_InSpecH.  AEGP\_SetInSpecHSF( AEIO\_InSpecH inH,  const A\_Ratio \*hsf); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetInSpecSoundRate | Obtains the sampling rate (in samples per second) for the audio data referenced by the AEIO\_InSpecH.  AEGP\_GetInSpecSoundRate( AEIO\_InSpecH inH,  A\_FpLong \*ratePF); |
| AEGP\_SetInSpecSoundRate | Sets the sampling rate (in samples per second) for the audio data referenced by the AEIO\_InSpecH.  AEGP\_SetInSpecSoundRate( AEIO\_InSpecH inH,  A\_FpLong rateF); |
| AEGP\_GetInSpecSoundEncoding | Obtains the encoding method (signed PCM, unsigned PCM, or floating point) from an AEIO\_InSpecH.  AEGP\_GetInSpecSoundEncoding( AEIO\_InSpecH inH, AEIO\_SndEncoding \*encodingP); |
| AEGP\_SetInSpecSoundEncoding | Sets the encoding method of an AEIO\_InSpecH.  AEGP\_SetInSpecSoundEncoding( AEIO\_InSpecH inH, AEIO\_SndEncoding encoding); |
| AEGP\_GetInSpecSoundSampleSize | Retrieves the bytes-per-sample (1,2, or 4) from an  AEIO\_InSpecH.  AEGP\_GetInSpecSoundSampleSize( AEIO\_InSpecH inH, AEIO\_SndSampleSize \*bytes\_per\_smpP); |
| AEGP\_SetInSpecSoundSampleSize | Set the bytes per sample of an AEIO\_InSpecH.  AEGP\_SetInSpecSoundSampleSize( AEIO\_InSpecH inH, AEIO\_SndSampleSize bytes\_per\_sample); |
| AEGP\_GetInSpecSoundChannels | Determines whether the audio in the  AEIO\_SndChannels is mono or stereo.  AEGP\_GetInSpecSoundChannels( AEIO\_InSpecH inH, AEIO\_SndChannels \*num\_channelsP); |
| AEGP\_SetInSpecSoundChannels | Sets the audio in an AEIO\_SndChannels to mono or stereo.  AEGP\_SetInSpecSoundChannels( AEIO\_InSpecH inH, AEIO\_SndChannels num\_channels); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_AddAuxExtMap | If your file format has auxiliary files which you want to  prevent users from opening directly, pass it’s extension, file type and creator to this function to keep it from appearing in input dialogs.  AEGP\_AddAuxExtMap(  const A\_char \*extension,  A\_long file\_type,  A\_long creator); |
| AEGP\_SetInSpecEmbeddedColor Profile | In case of RGB data, if there is an embedded icc profile, build an AEGP\_ColorProfile out of this icc profile using [AEGP\_GetNewColorProfileFromICCProfile](#_bookmark678) and set the profile description set to NULL.  In case of non-RGB data, if there is an embedded non-RGB icc profile or you know the color space the data is in, set the color profile set to NULL, and provide the description as a NULL-terminated unicode string. Doing this disables color management UI that allows user to affect profile choice in  the application UI.  If you are unpacking non-RGB data directly into working space (to get working space use [AEGP\_GetNewWorkingSpaceColorProfile](#_bookmark677)), you are done.  If you are unpacking non-RGB data into specific RGB color space, you must pass the profile describing this space to AEGP\_SetInSpecAssignedColorProfile. below. Otherwise, your RGB data will be incorrectly interpreted as being in working space.  Either color profile or profile description should be NULL in this function. You cannot use both.  AEGP\_SetInSpecEmbeddedColorProfile( AEIO\_InSpecH inH,  AEGP\_ConstColorProfileP  color\_profileP0, const A\_UTF16Char \*profile\_descP0); |
| AEGP\_SetInSpecAssignedColor Profile | Assign a valid RGB color profile to the footage.  AEGP\_SetInSpecAssignedColorProfile( AEIO\_InSpecH inH,  AEGP\_ConstColorProfileP  color\_profileP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetInSpecNativeStartTime | New in CC. Retrieves the native start time of the footage.  AEGP\_GetInSpecNativeStartTime( AEIO\_InSpecH inH,  A\_Time \*startTimeP); |
| AEGP\_SetInSpecNativeStartTime | New in CC. Assign a native start time to the footage.  AEGP\_SetInSpecNativeStartTime( AEIO\_InSpecH inH,  const A\_Time \*startTimeP); |
| AEGP\_ClearInSpecNativeStart- Time | New in CC. Clear the native start time of the footage. Setting the native start time to 0 using  AEGP\_SetInSpecNativeStartTime doesn't do this. It still means there is a special native start time provided.  AEGP\_ClearInSpecNativeStartTime( AEIO\_InSpecH inH); |
| AEGP\_GetInSpecNativeDisplay- DropFrame | New in CC. Retrieve the drop-frame setting of the footage.  AEGP\_GetInSpecNativeDisplayDropFrame( AEIO\_InSpecH inH,  A\_Boolean \*displayDropFrameBP); |
| AEGP\_SetInSpecNativeDisplay- DropFrame | New in CC. Assign the drop-frame setting of the footage.  AEGP\_SetInSpecNativeDisplayDropFrame( AEIO\_InSpecH inH,  A\_Boolean displayDropFrameB); |

These functions manage all interactions with an output specification in After Effects’ render queue.

**TABLE 114: AEGPIOOUTSUITE4**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetOutSpecOptionsHandle | Retrieves the Options for the AEIO\_OutSpecH.  AEGP\_GetOutSpecOptionsHandle( AEIO\_OutSpecH outH,  void \*\*optionsPPV); |
| AEGP\_SetOutSpecOptionsHandle | Sets the Options for the AEIO\_OutSpecH.  AEGP\_SetOutSpecOptionsHandle( AEIO\_OutSpecH outH,  void \*optionsPV, void \*\*old\_optionsPPV); |
| AEGP\_GetOutSpecFilePath | Obtains the path for the AEIO\_OutSpecH. The file path is a handle to a NULL-terminated A\_UTF16Char string, and must be disposed with AEGP\_FreeMemHandle.  If file\_rsrvdPB returns TRUE, the plug-in should not overwrite it (After Effects has already created an empty file); doing so can cause network renders to fail.  AEGP\_GetOutSpecFilePath( AEIO\_OutSpecH outH, AEGP\_MemHandle \*unicode\_pathPH, A\_Boolean \*file\_rsrvdPB);); |
| AEGP\_GetOutSpecFPS | Obtains the frames per second of the AEIO\_OutSpecH.  AEGP\_GetOutSpecFPS( AEIO\_OutSpecH outH,  A\_Fixed \*native\_fpsP); |
| AEGP\_SetOutSpecNativeFPS | Sets the frames per second of the AEIO\_OutSpecH.  AEGP\_SetOutSpecNativeFPS( AEIO\_OutSpecH outH, A\_Fixed native\_fpsP); |
| AEGP\_GetOutSpecDepth | Obtains the pixel bit depth of the AEIO\_OutSpecH.  AEGP\_GetOutSpecDepth( AEIO\_OutSpecH outH, A\_short \*depthPS); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_SetOutSpecDepth | Sets the pixel bit depth of the AEIO\_OutSpecH.  AEGP\_SetOutSpecDepth( AEIO\_OutSpecH outH, A\_short depthPS); |
| AEGP\_GetOutSpecInterlaceLabel | Obtains field information for the AEIO\_OutSpecH.  AEGP\_GetOutSpecInterlaceLabel( AEIO\_OutSpecH outH, FIEL\_Label \*interlaceP); |
| AEGP\_SetOutSpecInterlaceLabel | Set the field information for the AEIO\_OutSpecH.  AEGP\_SetOutSpecInterlaceLabel( AEIO\_OutSpecH outH,  const FIEL\_Label \*interlaceP); |
| AEGP\_GetOutSpecAlphaLabel | Obtains alpha interpretation information for the  AEIO\_OutSpecH.  AEGP\_GetOutSpecAlphaLabel( AEIO\_OutSpecH outH, AEIO\_AlphaLabel \*alphaP); |
| AEGP\_SetOutSpecAlphaLabel | Sets the alpha interpretation for the AEIO\_OutSpecH.  AEGP\_SetOutSpecAlphaLabel( AEIO\_OutSpecH outH, const AEIO\_AlphaLabel \*alphaP); |
| AEGP\_GetOutSpecDuration | Obtains the duration of the AEIO\_OutSpecH.  AEGP\_GetOutSpecDuration( AEIO\_OutSpecH outH,  A\_Time \*durationP); |
| AEGP\_SetOutSpecDuration | Sets the duration of the AEIO\_OutSpecH.  AEGP\_SetOutSpecDuration( AEIO\_OutSpecH outH,  const A\_Time \*durationP); |
| AEGP\_GetOutSpecDimensions | Obtains the dimensions of the AEIO\_OutSpecH.  AEGP\_GetOutSpecDimensions( AEIO\_OutSpecH outH, A\_long \*widthPL0,  A\_long \*heightPL0); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetOutSpecHSF | Obtains the horizontal scaling factor of the  AEIO\_OutSpecH.  AEGP\_GetOutSpecHSF( AEIO\_OutSpecH outH, A\_Ratio \*hsf); |
| AEGP\_SetOutSpecHSF | Sets the horizontal scaling factor of the AEIO\_OutSpecH.  AEGP\_SetOutSpecHSF( AEIO\_OutSpecH outH, const A\_Ratio \*hsf); |
| AEGP\_GetOutSpecSoundRate | Obtains the sampling rate for the AEIO\_OutSpecH.  AEGP\_GetOutSpecSoundRate( AEIO\_OutSpecH outH, A\_FpLong \*ratePF); |
| AEGP\_SetOutSpecSoundRate | Sets the sampling rate for the AEIO\_OutSpecH.  AEGP\_SetOutSpecSoundRate( AEIO\_OutSpecH outH, A\_FpLong rateF); |
| AEGP\_GetOutSpecSoundEncoding | Obtains the sound encoding format of the  AEIO\_OutSpecH.  AEGP\_GetOutSpecSoundEncoding( AEIO\_OutSpecH outH, AEIO\_SndEncoding \*encodingP); |
| AEGP\_SetOutSpecSoundEncoding | Sets the sound encoding format of the AEIO\_OutSpecH.  AEGP\_SetOutSpecSoundEncoding( AEIO\_OutSpecH outH, AEIO\_SndEncoding encoding); |
| AEGP\_GetOutSpecSoundSample Size | Obtains the bytes-per-sample of the AEIO\_OutSpecH.  AEGP\_GetOutSpecSoundSampleSize( AEIO\_OutSpecH outH, AEIO\_SndSampleSize \*bpsP); |
| AEGP\_SetOutSpecSoundSample Size | Sets the bytes-per-sample of the AEIO\_OutSpecH.  AEGP\_SetOutSpecSoundSampleSize( AEIO\_OutSpecH outH, AEIO\_SndSampleSize bpsP); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetOutSpecSoundChannels | Obtains the number of sounds channels in the  AEIO\_OutSpecH.  AEGP\_GetOutSpecSoundChannels( AEIO\_OutSpecH outH, AEIO\_SndChannels \*channelsP); |
| AEGP\_SetOutSpecSoundChannels | Sets the number of sounds channels in the  AEIO\_OutSpecH.  AEGP\_SetOutSpecSoundChannels( AEIO\_OutSpecH outH, AEIO\_SndChannels channels); |
| AEGP\_GetOutSpecIsStill | Determines whether the AEIO\_OutSpecH is a still.  AEGP\_GetOutSpecIsStill( AEIO\_OutSpecH outH, A\_Boolean \*is\_stillPB); |
| AEGP\_GetOutSpecPosterTime | Obtains the time of the AEIO\_OutSpecH’s poster frame.  AEGP\_GetOutSpecPosterTime( AEIO\_OutSpecH outH,  A\_Time \*poster\_timeP); |
| AEGP\_GetOutSpecStartFrame | Obtains the time of the first frame in the  AEIO\_OutSpecH.  AEGP\_GetOutSpecStartFrame( AEIO\_OutSpecH outH,  A\_long \*start\_frameP); |
| AEGP\_GetOutSpecPullDown | Obtains the pulldown phase of the AEIO\_OutSpecH.  AEGP\_GetOutSpecPullDown( AEIO\_OutSpecH outH, AEIO\_Pulldown \*pulldownP); |
| AEGP\_GetOutSpecIsMissing | Passes back TRUE if there is no AEIO\_OutSpecH.  AEGP\_GetOutSpecIsMissing( AEIO\_OutSpecH outH, A\_Boolean \*missingPB); |
| AEGP\_GetOutSpecShouldEmbedICC Profile | Returns TRUE if the AEIO should embed a color profile in the output.  AEGP\_GetOutSpecShouldEmbedICCProfile( AEIO\_OutSpecH outH,  A\_Boolean \*embedPB); |

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| AEGP\_GetNewOutSpecColor Profile | Returns an (opaque) ICC color profile for embedding in the AEIO’s output. Must be disposed with AEGP\_DisposeColorProfile.  AEGP\_GetNewOutSpecColorProfile( AEGP\_PluginID aegp\_plugin\_id, AEIO\_OutSpecH outH, AEGP\_ColorProfileP  \*color\_profilePP); |
| AEGP\_GetOutSpecOutputModule | Returns the AEGP\_RQItemRefH and AEGP\_OutputModuleRefH associated with the given AEIO\_OutSpecH. Fails if the render queue item is not  found, or if AEIO\_OutSpecH is not a confirmed outH and is a copy, i.e. if the Output Module settings dialog is open  and the user hasn’t hit OK.  AEGP\_GetOutSpecOutputModule( AEIO\_OutSpecH outH, AEGP\_RQItemRefH \*rq\_itemP, AEGP\_OutputModuleRefH  \*om\_refP); |

### IMPLEMENTATION DETAILS

**EXPORT BIT-DEPTH**

In the Output Module Settings, the user can choose a Depth based on the options the AEIO declares support for in [AEIO\_GetDepths()](#_bookmark802). If a plug-in supports higher bit-depth

exports, it should be able to handle these higher bit-depth PF\_EffectWorlds passed in [AEIO\_AddFrame()](#_bookmark796) or [AEIO\_OutputFrame()](#_bookmark798), even when the export setting is not set to the same depth. The frame delivered to the AEIO, and the final output will not necessarily be the same depth. You may get frames passed in the project bit-depth instead of the final

output if After Effects thinks that will be higher quality.

**USER DATA VS. OPTIONS**

It’s possible to use either user data allocations or options handles to store metadata about a file. We use user data for information that’s to be embedded in the file (presuming the file

format supports such information); marker data, field labels, etc. We use option handles for information about the file; output settings, dimensions, details of compression settings used.

10 : PREMIERE PRO &

OTHER HOSTS

Adobe Premiere Pro and Adobe Premiere Elements both support the After Effects effect API as described in chapters 2, 3, and 5. They offer a thorough host implementation, some

the key omissions being 3D-related calls (auxiliary channel information, cameras and lights), 16-bit and SmartFX support, and other utility functions provided by After Effects’ AEGP API.

Both Premiere Pro and Premiere Elements set PF\_InData>appl\_id to ‘PrMr’. In this chapter, we will describe the AE API support in Premiere Pro, but generally the same

support exists in corresponding versions of Premiere Elements. If you need to distinguish

between Premiere Pro and Premiere Elements, you may use the Premiere-specific App Info Suite, available from the [Premiere Pro SDK](http://www.adobe.com/devnet/premiere.html) headers.

|  |  |  |
| --- | --- | --- |
| **Application Versions** | **PF\_InData> version.major** | **PF\_InData> version.minor** |
| Premiere Pro CC and Premiere Pro CC 2014 | 13 | 4 |
| Premiere Pro CS6 | 13 | 2 |
| Premiere Pro CS5.5 | 13 | 1 |
| Premiere Pro CS5, Premiere Elements 9 | 13 | 0 |
| Premiere Pro CS4, Premiere Elements 8 | 12 | 5 |
| Premiere Pro CS3, Premiere Elements 4 and 7 | 12 | 4 |
| Premiere Pro 2.0, Premiere Elements 3 | 12 | 3 |
| Premiere Pro 1.5, Premiere Elements 2 | 12 | 2 |
| Premiere Pro 1.0, Premiere Elements 1 | 12 | 1 |

Note that the versioning used by Premiere Pro and Premiere Elements does not mean that

they support the same API features After Effects did at the same version. It is simply meant to distinguish from one version to the next.

### PLUG-IN INSTALLATION

Use the common plug-in folder as described [here](#_bookmark57). If you try to install an effect plug-in only to the Premiere Pro plug-ins directory, you will be surprised to find that your effect is not rendered when you export to disk through Adobe Media Encoder, an entirely separate

application. Oh, and you’ll also miss out on project interchange and copy / paste between Premiere Pro and After Effects.

**PLUG-INS... RELOADED**

On it’s first launch, Premiere Pro loads all the plug-ins, reads the PiPL, and sends

PF\_Cmd\_GLOBAL\_SETUP to determine the plug-ins’ capabilities. To save time on future

application launches, it saves some of these capabilities in what we call the plug-in cache (the registry on Windows, a Property List file on Mac OS). The next time the application is

launched, the cached information is used wherever possible, rather than loading the plug-ins.

When debugging, you can always force a reload of all the plug-ins by holding down the Shift key when launching Premiere Pro.

If your effect needs to be reloaded each time, there is a way to disable this caching. The plug- in can use the PF Cache On Load Suite in AE\_CacheOnLoadSuite.h (from the [Premiere](http://www.adobe.com/devnet/premiere.html)

[Pro SDK](http://www.adobe.com/devnet/premiere.html) headers) to call PF\_SetNoCacheOnLoad() during PF\_Cmd\_GLOBAL\_SETUP. For the second parameter of that function, pass a non-zero value if you want your effect to show up in the UI. Pass zero if loading failed, but you still want Premiere Pro to attempt to load it again on the next relaunch.

### BASIC HOST DIFFERENCES

We’ve tried to provide robust compatibility for After Effects effect plug-ins in Premiere Pro.

There are underlying differences in the render pipeline that lead to differences, and we

realize the API implementation may not be perfect. If you run into anomalous or just plain confusing behavior, please [let us know](mailto:zlam@adobe.com). Below is an overview of some differences you will

notice the first time you install your AE effect into PPro and apply it to a clip.

**TIME**

Premiere Pro uses slightly different time values in PF\_InData. For example in CS4:

Rendering in NTSC, time\_scale is 60000, time\_step is 1001, field gives field order (in After Effects, for field rendering, scale is 2997, step is 50, or for progressive rendering, scale is 2997, step is 100).

Rendering in PAL, time\_scale is 50, time\_step is 1, field gives field order (in After

Effects, for field rendering, scale is 3200, step is 64, or for progressive rendering, scale is 3200, step is 128.

It's the ratio of time-related values that produces the time value, not specifically the time\_scale value. It's possible Premiere Pro will use different time\_scales in the future, so please don't hard code. Just be aware that it does not necessarily use the exact same values as After Effects.

**FRAME DIMENSIONS**

Differences between source footage and the project/composition are handled differently. For example, in CS4, when importing an NTSC clip in a PAL sequence,

PF\_InData>width,height are (598,480) and PF\_InData->pixel\_aspect\_ratio is (768,702). In AE, width,height are (720,480) and pixel\_aspect\_ratio is (10,11).

**PF\_INDATA**

Premiere Pro handles field rendering differently than After Effects. While field rendering, PF\_InData>field gives the current field being rendered, ignoring whether or not PF\_OutFlag\_PIX\_INDEPENDENT flag was set.

In Premiere Pro, effects receive the quality setting of the monitor window in [PF\_InData>quality](#_bookmark114). This differs from After Effects, where the source layer’s quality setting is provided here.

**PARAMETER UI**

Premiere Pro does not honor the [PF\_ParamFlag\_START\_COLLAPSED](#_bookmark221) flag. Parameters

are always initialized with their twirlies collapsed, and cannot be automatically twirled open by parameter supervision.

**MISSING SUITES**

Many suites supported by After Effects are not implemented in the Premiere Pro host. In several cases, even if a suite is missing in Premiere Pro, an equivalent macro function is

available. Here are a few examples:

|  |  |
| --- | --- |
| **After Effects suite call** | **Premiere Pro equivalent function** |
| WorldTransformSuite1()->copy() | PF\_COPY() |
| WorldTransformSuite1()->convolve() | in\_data->utils->convolve() |
| FillMatteSuite2()->fill() | PF\_FILL() |
| PF\_PixelDataSuite1->get\_pixel\_data8() | PF\_GET\_PIXEL\_DATA8() |

The sample projects demonstrate alternate ways of handling a missing suite, by checking for the host application and version. The Portable sample project demonstrates both host

application and version checking.

**A SPECIAL SUITE FOR AE EFFECTS RUNNING IN PREMIERE PRO**

No AEGP calls are supported by Premiere Pro. However, there are some interesting

parallels in the header PrSDKAESupport.h. For example, you can use the Utility Suite in that header to get the frame rate or field type of the source footage, or to get the speed

applied to the clip.

Note that other suites from the Premiere Pro SDK cannot be used in AE effects.

### MULTITHREADING

Premiere Pro was built to provide real-time playback of footage with effects wherever

possible. The render scheduling is much more aggressive and multithreaded rendering is a

basic requirement. This is quite different than After Effects, where users are building layers upon layers of effects and more willing to wait for a RAM preview.

Multithreaded rendering in Premiere is extended to AE effects. When rendering an AE effect, the messaging from Premiere passes through a critical section which is used for all

commands, except those relating to arbitrary data. The critical section prevents two threads from calling the same instance of the effect at the same time. However, Premiere creates

multiple instances of the effect, which can be called concurrently from separate threads.

Therefore, an effect should not expect to receive render requests in order of increasing time. Also, effects should not depend on static, non-constant variables.

A flag has been added to the AE API, PF\_OutFlag2\_PPRO\_DO\_NOT\_CLONE\_SEQUENCE\_DATA\_FOR\_RENDER, to allow a plug-in to opt out of Premiere Pro's multithreaded rendering of AE effects. We advise against setting this flag, as it has been found to cause parameter UI problems.

When the flag is set, we don't clone the sequence data across all the threads and we only call into the plug-in on one thread at a time. Premiere Pro will still render using multiple

threads, but the effect will only render on one thread at a time, and the same sequence data will be used. This flag is useful for plug-ins that provide their own internal multithreading, or plug-ins that render frames based on previous frames, such as image stabilizers.

### BIGGER DIFFERENCES

As long as an effect only supports the basic ARGB\_8u pixel format supported by After

Effects, Premiere Pro will try to imitate the After Effects hosting behavior and hide various differences because of the different render pipeline architecture. But if an effect wants to

support additional pixel formats, such as 32-bit RGB, be prepared to handle further divergent behavior.

When rendering native pixel formats, Premiere will send PF\_Cmd\_RENDER once for each field, rather than for each frame. The PF\_InData->field will indicate which field is being rendered, the PF\_LayerDef->height will be half of the frame height, and the PF\_LayerDef->rowbytes will be double the normal value.

**PIXEL FORMATS**

Premiere Pro provides function suites for declaring support for pixel formats other than the 8-bit RGB format used by After Effects - ARGB\_8u. These pixel formats include the

Premiere Pro native 8-bit RGB format - BGRA\_8u, as well as YUV, 32-bit formats, and

more. For a more detailed discussion of the various pixel formats, see the [Premiere Pro SDK](http://www.adobe.com/devnet/premiere.html) Guide, chapter 3, in the section “Pixel Formats and Colorspaces”.

Use the PF Pixel Format Suite (defined in PrAESDKSupport.h) to register for [PF\_EffectWorlds](#_bookmark228) in other pixel formats. Use the Premiere Pixel Format Suite (defined in the aptly-named PrSDKPixelFormatSuite.h) to get black and white values in those pixel formats.

After Effects functions such as PF\_BLEND() have not been enhanced to work with pixel formats beyond 8-bit RGB.

**32-BIT FLOAT SUPPORT**

Premiere Pro does not support After Effects 16-bit rendering or SmartFX. For 32-bit

rendering in Premiere Pro, you’ll need to declare support for one of the 32-bit pixel formats (see previous section), and then implement 32-bit rendering for PF\_Cmd\_RENDER. You can

support multiple render depths this way. See the SDK Noise sample project for an example.

Depending on the clip(s) to which an effect is applied, 32-bit processing is not always

necessary to preserve the quality of the source input. But there are settings to force 32-bit rendering, to give effects processing finer granularity and more headroom, if desired. Go to Settings>Sequence Settings> Video Previews>Maximum Bit Depth, to control previewing

from the timeline. For export to file, use Export Settings>Video>Basic Settings>Render at Maximum Depth.

**PF\_CHECKOUT\_PARAM AND PIXEL FORMATS**

Before CS6, PF\_CHECKOUT\_PARAM() only returned 8-bit ARGB buffers, regardless of the pixel format currently being used for rendering. Starting in CS6, an effect can opt in to get frames in the same format as the render request, whether it is 32-bit float, YUV, etc.

Plug-ins may request this behavior, but existing plug-ins will continue working receiving 8- bit ARGB frames. The call is EffectWantsCheckedOutFramesToMatch RenderPixelFormat(), in the PF Utility Suite, defined in PrSDKAESupport.h. The call should be made on PF\_Cmd\_GLOBAL\_SETUP, the same selector where an effect would already advertise support beyond 8-bit RGB using AddSupportedPixelFormat().

**EFFECTS PRESETS**

Premiere Pro uses a different preset scheme than After Effects. From the Premiere Pro SDK Guide:

Effect presets appear in the Presets bin in the Effects panel, and can be applied just like Effects with specific parameter settings and keyframes. Effect presets can be created as follows:

1. Apply a filter to a clip
2. Set the parameters of the filter, adding keyframes if desired
3. Right-click on the filter name in the Effect Controls panel, and select “Save Preset...”
4. Create preset bins if desired by right-clicking in the Effects panel and choosing “New Presets Bin”
5. Organize the presets in the preset folders
6. Select the bins and/or presets you wish to export, right-click, and choose “Export Preset”

Presets should be installed in the Plug-ins directory. Once they are installed in that

directory, they will be read-only, and the user will not be able to move them to a different folder or change their names. User-created presets will be modifiable. On Windows Vista, these are in the user’s hidden AppData folder (e.g. C:\Users\[user name]\AppData\Roaming\Adobe\Premiere Pro\[version]\Effect Presets and Custom Items.prfpset). On Mac OS, they are in the user folder, at ~/Library/ Application Support/Adobe/Premiere Pro/[version]/Effect Presets and Custom Items.prfpset.

**CUSTOM ECW UI OVER A STANDARD DATA TYPE**

While this is logged as bug #1235407, there is a simple workaround: Create two separate parameters, and have the custom UI control the slider param using parameter supervision.

### PREMIERE ELEMENTS

Premiere Elements (but not Premiere Pro) displays visual icons for each effect. You will need to provide icons for your effects, or else an empty black icon will be shown for your

effects, or even worse behavior in Premiere Elements 8. The icons are 60x45 PNG files, and are placed here:

[Program Files]\Adobe\Adobe Premiere Elements [version]\Plug- ins\Common\EffectPreviews\

The filename should be the match name of the effect, which you specify in the [PiPL](#_bookmark49),

prefixed with “AE.” So if the match name was “MatchName”, then the filename should be “AE.MatchName.png”

### UNSUPPORTED FEATURES

Premiere Pro is currently known to not support the following features of the After Effects API:

(If you would like a feature with a "-" bullet, please email [Premiere Pro API Engineering](mailto:zlam@adobe.com) with the feature request. Numbers preceded by an ‘F’ are feature request numbers, and the others are bug numbers)

F7233 - extent\_hint support

F7835 - Multiple PiPLs in a single plug-in F7836 - AEGP support

F7517 - Audio support - if a plug-in sets PF\_OutFlag\_I\_USE\_AUDIO in

PF\_Cmd\_GLOBAL\_SETUP, it will not be loaded at all

F9355 - Support PF\_ParamFlag\_COLLAPSE\_TWIRLY

3689851 - PF\_OutFlag\_I\_AM\_OBSOLETE unsupported

* PF World Transform Suite
* PF AE Channel Suite
* AE’s implementation of high bit color depth support
* SmartFX
* 3D support
* PF\_SUBPIXEL\_SAMPLE(), PF\_GET\_PIXEL\_DATA16()

**BUT...WHY’D YOU LOAD IT, IF YOU CAN’T RUN IT?!**

Premiere Pro attempts to load AEGP plug-ins. To detect this and avoid any problem

behavior, your command hook function can access a suite which is only provided by After Effects; AEGP\_CanvasSuite is a fine candidate. If the suite isn’t present, return an error. The plug-in will be placed on Premiere Pro’s “don’t load these” list.

**OTHER HOSTS?**

For third-party hosts, the Adobe policy remains:

*“Adobe neither supports nor recommends the creation of Adobe-compatible third-party hosts.*

*While it may be possible to create a partially functional host by reverse engineering from the plug- in API specification, we do not recommend it and will not support you in doing so.”*

**REALITY SANDWICH**

We realize that, for developers like you, one good way to grow your market is to ensure that your plug-ins work in as many hosts as possible. Our SmartFX API has created quite a bit of distance between the After Effects API and the implementations available in the rest of the plug-in hosting world. We will do what we can to help the other hosts support newer

features. If you encounter problems in third party hosts, please refer them to us if they need assistance.

[Version History 2](#_bookmark0)

[Coding conventions 5](#_bookmark5)

[Sample Project Descriptions 38](#_bookmark39)

[API Versions 42](#_bookmark46)

[Entry Point Function Parameters 48](#_bookmark65)

[Command Selectors 50](#_bookmark72)

[PF\_InData. 56](#_bookmark112)

[PF\_OutData 61](#_bookmark129)

[PF\_OutFlags 62](#_bookmark144)

[PF\_OutFlags2. 66](#_bookmark168)

[Parameter Types 70](#_bookmark188)

[PF\_ParamDef 74](#_bookmark210)

[Parameter UI Flags 75](#_bookmark213)

[Parameter Flags 77](#_bookmark217)

[PF\_EffectWorld structure 79](#_bookmark229)

[PF\_PixelPtr accessor macros 81](#_bookmark243)

[Error Codes 82](#_bookmark247)

[PF\_HandleSuite1 86](#_bookmark259)

[PF\_WorldSuite2 87](#_bookmark261)

[PF\_Iterate8Suite1, PF\_Iterate16Suite1, PF\_IterateFloatSuite1 88](#_bookmark263)

[PF\_WorldTransformSuite1 91](#_bookmark266)

[Kernel Flags 94](#_bookmark270)

[PF\_FillMatteSuite2 95](#_bookmark272)

[PF\_SamplingSuite Functions (multiple suites) 96](#_bookmark274)

[PF\_BatchSamplingSuite1 97](#_bookmark275)

[PF\_ANSICallbacksSuite1 98](#_bookmark278)

[Interaction Callbacks 100](#_bookmark282)

[PF\_ColorParamSuite1 108](#_bookmark308)

[PF\_PointParamSuite1 108](#_bookmark310)

[PF\_AngleParamSuite1 108](#_bookmark312)

[PF\_ParamUtilsSuite3 110](#_bookmark319)

[Arbitrary data selectors 116](#_bookmark336)

[PF\_EffectUISuite. 119](#_bookmark347)

[PF\_AppSuite5 120](#_bookmark350)

[AE\_AdvAppSuite2 122](#_bookmark354)

[PF\_AdvTimeSuite2 124](#_bookmark356)

[PF\_AdvItemSuite1 126](#_bookmark358)

[PF\_ChannelSuite1 127](#_bookmark360)

[PF\_PathVertex 130](#_bookmark367)

[PF\_PathDataSuite1. 130](#_bookmark370)

[PF\_PathQuerySuite 133](#_bookmark373)

[Pixel Types for different color spaces 135](#_bookmark376)

[color space conversion callbacks 135](#_bookmark381)

[PF\_PreRenderExtra 145](#_bookmark406)

[PF\_PreRenderOutput 148](#_bookmark410)

[PF\_SmartRenderExtra 150](#_bookmark416)

[Events 153](#_bookmark421)

[PF\_EventExtra 154](#_bookmark436)

[PF\_Context 156](#_bookmark445)

[PF\_EffectWindowInfo 156](#_bookmark446)

[PF\_DoClickEventInfo 157](#_bookmark456)

[PF\_DrawEventInfo 157](#_bookmark467)

[PF\_KeyDownEvent 158](#_bookmark472)

[PF\_AdjustCursorEventInfo 159](#_bookmark474)

[PF\_ArbParamsExtra 159](#_bookmark476)

[PF\_EffectCustomUISuite1 161](#_bookmark483)

[DRAWBOT\_DrawbotSuite1 162](#_bookmark486)

[DRAWBOT\_SupplierSuite1 162](#_bookmark488)

[DRAWBOT\_SurfaceSuite1 164](#_bookmark491)

[DRAWBOT\_PathSuite1 167](#_bookmark494)

[PF\_EffectCustomUIOverlayThemeSuite1 168](#_bookmark497)

[UI Callbacks 170](#_bookmark499)

[Audio data structures 174](#_bookmark513)

[AEGP API Data Types 179](#_bookmark533)

[Data types requiring disposal 181](#_bookmark536)

[AEGP Suites 184](#_bookmark546)

[AEGP\_MemorySuite1 187](#_bookmark550)

[AEGP\_CommandSuite1 188](#_bookmark552)

[AEGP\_RegisterSuite5 191](#_bookmark557)

[AEGP\_ProjSuite6 193](#_bookmark563)

[AEGP\_TimeDisplay2 195](#_bookmark565)

[AEGP\_ItemSuite8 196](#_bookmark568)

[AEGP\_CollectionSuite2 202](#_bookmark571)

[AEGP\_CompSuite10 203](#_bookmark575)

[AEGP\_FootageSuite5 210](#_bookmark582)

[AEGP\_FootageInterp structure 217](#_bookmark589)

[AEGP\_LayerSuite8 218](#_bookmark592)

[AEGP\_EffectSuite4 228](#_bookmark608)

[AEGP\_StreamSuite4 235](#_bookmark622)

[AEGP\_DynamicStreamSuite4 242](#_bookmark633)

[AEGP\_KeyframeSuite3 250](#_bookmark641)

[AEGP\_MarkerSuite2 255](#_bookmark650)

[AEGP\_MaskSuite6 258](#_bookmark653)

[AEGP\_MaskOutlineSuite3 261](#_bookmark659)

[AEGP\_TextDocumentSuite1 264](#_bookmark662)

[AEGP\_TextLayerSuite1 265](#_bookmark664)

[AEGP\_UtilitySuite6 266](#_bookmark666)

[AEGP\_PersistentDataSuite4 271](#_bookmark674)

[AEGP\_ColorSettingsSuite2 275](#_bookmark676)

[AEGP\_RenderOptionsSuite4 277](#_bookmark681)

[AEGP\_LayerRenderOptionsSuite1 (New in 13.0) 281](#_bookmark682)

[AEGP\_RenderSuite4 284](#_bookmark684)

[AEGP\_WorldSuite3 288](#_bookmark690)

[AEGP\_CompositeSuite2 291](#_bookmark694)

[AEGP\_SoundDataSuite1 293](#_bookmark696)

[AEGP\_RenderQueueSuite1 294](#_bookmark699)

[AEGP\_RQItemSuite3 295](#_bookmark702)

[AEGP\_RenderQueueMonitorSuite1 297](#_bookmark705)

[AEGP\_OutputModuleSuite4 302](#_bookmark707)

[AEGP\_PFInterfaceSuite1 307](#_bookmark710)

[AEGP\_IterateSuite1 308](#_bookmark714)

[AEGP\_FIMSuite3 309](#_bookmark716)

[Data types used in the Artisan API 312](#_bookmark730)

[Artisan Entry Points 314](#_bookmark736)

[AEGP\_CanvasSuite8 317](#_bookmark741)

[AEGP\_ArtisanUtilSuite1 326](#_bookmark748)

[AEGP\_CameraSuite2 327](#_bookmark750)

[AEGP\_LightSuite2 329](#_bookmark756)

[AEGP\_QueryXformSuite2 330](#_bookmark761)

[PR\_InteractiveDrawProcs 333](#_bookmark763)

[AEIO\_ModuleInfo 338](#_bookmark778)

[AEIO\_ModuleFlags 339](#_bookmark780)

[AEIO\_ModuleFlags2 341](#_bookmark782)

[AEIO\_FunctionBlock4 342](#_bookmark784)

[AEGP\_IOInSuite5 354](#_bookmark808)

[AEGPIOOutSuite4 360](#_bookmark812)

A

[AEFX\_CLR\_STRUCT 70](#_bookmark187)

[AEGP\_StreamSuite1 235](#_bookmark621)

[area 156](#_bookmark448)

[Audio-specific Float Slider Variables 175](#_bookmark516)

C

[cbs 155](#_bookmark438)

[cmd 48](#_bookmark66)

[continue\_refcon 157](#_bookmark461)

[current\_frame 156](#_bookmark451)

D

[data 79](#_bookmark231)

[Debugging 176](#_bookmark523)

[depth 157](#_bookmark469)

[Draw Event 157](#_bookmark465)

E

[effect\_win 155](#_bookmark437)

[evt\_in\_flags 155](#_bookmark439)

[evt\_out\_flags 155](#_bookmark441)

[extent\_hint 79](#_bookmark237)

F

[flat\_sdata\_size 61](#_bookmark134)

[frame\_data 61](#_bookmark135)

G

[GET\_LAYER2COMP\_XFORM 172](#_bookmark502)

[global\_data 61](#_bookmark132)

[Graphics Utility Callbacks 98](#_bookmark277)

H

[height . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61,](#_bookmark136) [79](#_bookmark235)

[HLS pixel 135](#_bookmark378)

[horiz\_offset 156](#_bookmark453)

I

[in\_data 48](#_bookmark67)

[index 156](#_bookmark447)

K

[Key Down Event 158](#_bookmark471)

L

[last\_time 157](#_bookmark463)

[luminance 136](#_bookmark382)

M

[modifiers 157](#_bookmark460)

[my\_version 61](#_bookmark130)

N

[name 61](#_bookmark131)

[num\_clicks 157](#_bookmark459)

[num\_params 61](#_bookmark133)

O

[origin 61](#_bookmark137)

[out\_data 49](#_bookmark69)

[out\_flags 61](#_bookmark139)

P

[param\_title\_frame 156](#_bookmark452)

[Parameter supervision 108](#_bookmark314)

[Parameters 61](#_bookmark128)

[PF\_ABORT 100](#_bookmark284)

[PF\_ADD\_PARAM 69](#_bookmark185)

[PF\_Arbitrary\_FLAT\_SIZE\_FUNC 116](#_bookmark338)

[PF\_Arbitrary\_INTERP\_FUNC 117](#_bookmark339)

[PF\_Arbitrary\_PRINT\_SIZE\_FUNC 117](#_bookmark340)

[PF\_CHECKIN\_LAYER\_AUDIO 103](#_bookmark295)

[PF\_CHECKIN\_PARAM 102](#_bookmark291)

[PF\_CHECKOUT\_LAYER\_AUDIO 102](#_bookmark294)

[PF\_CHECKOUT\_PARAM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101,](#_bookmark288) [142](#_bookmark397)

[PF\_Cmd\_ABOUT 48](#_bookmark64)

[PF\_Cmd\_GLOBAL\_SETUP 63](#_bookmark149)

[PF\_Cmd\_PARAMS\_SETUP 69](#_bookmark186)

[PF\_DrawEventInfo 157](#_bookmark466)

[PF\_EA\_CONTROL 156](#_bookmark449)

[PF\_EA\_PARAM\_TITLE 156](#_bookmark450)

[PF\_Event\_ACTIVATE 153](#_bookmark423)

[PF\_Event\_ADJUST\_CURSOR 154](#_bookmark431)

[PF\_Event\_CLOSE\_CONTEXT 154](#_bookmark429)

[PF\_Event\_DEACTIVATE 154](#_bookmark428)

[PF\_Event\_DO\_CLICK 153](#_bookmark424)

[PF\_Event\_DRAG 154](#_bookmark426)

[PF\_Event\_DRAW 154](#_bookmark427)

[PF\_Event\_IDLE 154](#_bookmark430)

[PF\_Event\_KEYDOWN 154](#_bookmark432)

[PF\_Event\_NEW\_CONTEXT 153](#_bookmark422)

[PF\_ExtDependenciesExtra 54](#_bookmark107)

[PF\_GET\_AUDIO\_DATA 103](#_bookmark296)

[PF\_HLS\_PIXEL 135](#_bookmark379)

[PF\_InData 100](#_bookmark281)

[PF\_InData Structure 56](#_bookmark111)

[PF\_OutData structure 61](#_bookmark128)

[PF\_Param\_ANGLE 71](#_bookmark193)

[PF\_Param\_ARBITRARY 72](#_bookmark198)

[PF\_Param\_BUTTON 73](#_bookmark202)

[PF\_Param\_CHECKBOX 71](#_bookmark194)

[PF\_Param\_COLOR 71](#_bookmark195)

[PF\_Param\_FLOAT\_SLIDER 70](#_bookmark192)

[PF\_Param\_GROUP\_END 72](#_bookmark201)

[PF\_Param\_GROUP\_START 72](#_bookmark200)

[PF\_Param\_LAYER 70](#_bookmark189)

[PF\_Param\_PATH 72](#_bookmark199)

[PF\_Param\_POINT\_3D 73](#_bookmark203)

[PF\_Param\_POPUP 71](#_bookmark197)

[PF\_Param\_SLIDER 70](#_bookmark190)

[PF\_ParamDef . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101,](#_bookmark290) [102](#_bookmark292)

[PF\_Pixel 135](#_bookmark377)

[PF\_PlatData\_RES\_FILE\_PATH 172](#_bookmark501)

[PF\_PROGRESS 101](#_bookmark286)

[PF\_REGISTER\_UI 102](#_bookmark293)

[PF\_YIO\_PIXEL 135](#_bookmark380)

[platform\_ref 80](#_bookmark238)

R

[return\_msg . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61,](#_bookmark141) [63](#_bookmark151)

[rowbytes 79](#_bookmark233)

S

[screen\_point 157](#_bookmark458)

[send\_drag 157](#_bookmark462)

U

[update\_rect 157](#_bookmark468)

W

[when 157](#_bookmark457)

[width . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61,](#_bookmark138) [79](#_bookmark234)

[world\_flags 79](#_bookmark230)