

After Effects CS5 SDK Guide

Release 1

April 28, 2010

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.

TABLE 1: VERSION HISTORY

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.

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 a decade 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](#) 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](#). 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](#) 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 stable.

[SmartFX](#) 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](#) 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. NOTE: there have been many changes (we like to call them improvements) to custom UI messaging for 7.0; test your plug-ins carefully, and feel free to ask us questions.

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

Chapter 7 details the After Effects General Plug-in ([AEGP](#)) 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](#), specialized plug-in 3D renderer AEGPs, are covered in chapter 8. Under no circumstances should anyone write an artisan, ever.

Chapter 9 documents [AEIOs](#), 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
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
About this document	3
Organization	4
Documentation Conventions	4
A note about coding style	5
Table 2: Coding conventions	5

CHAPTER 1: INTRODUCTION 22

SDK Audience	22
Development Requirements	22
What plug-ins are	23
Where plug-ins appear in After Effects	23
How After Effects interacts with plug-ins	23
SDK contents	24
Other integration possibilities	24
Scripting	24
Pixel Bender Filters	24
Premiere Pro Importers	25
Hardware Output APIs	25
What's new?	25
What's new in CS5 (10.0)?	25
Do I Have to Update my Plug-ins for CS5?	26
Quick Tips for 64-bit Windows Porting	26
Quick Tips for 64-bit Mac OS Porting	27
What About Backwards Compatibility?	28
What was new in CS4 (9.0)?	28
What was new in CS3 (8.0)?	29
...and what was new before CS3?	29
Why develop After Effects plug-ins?	29
Developers matter	30

Sample projects	30
Table 3: Sample Project Descriptions	30
How to start creating plug-ins	32
Play!	32
Plan!	33
Hack!	33
Steal!	33
Test!	33
Debugging plug-ins	34
Slick debugging tricks	34
Where plug-ins are installed	34
Do I have to install the plug-ins to the common folder?	35
Compatibility across multiple versions?	35
Table 4: Effect API Versions	35
Third-party plug-in hosts?	36
PiPL resources	36
Entry Point	37
PiPL resources and Microsoft Visual Studio	37
Multiple PiPLs	37
Super Secret PiPL bit	37
Why do I need to know all this?	38
Exceptions	38
Localization	38
Next steps	38

CHAPTER 2: EFFECT BASICS 39

Entry Point	39
Table 5: Effect plug-in entry point	39
Command Selectors	40
Table 6: Command Selectors	40
Responding to selectors	44

Calling sequence	45
What's the difference?	45
PF_InData	46
Table 7: PF_InData	46
extent_hint usage	49
Now with 20% more pixels!	50
Point controls and buffer expansion	50
PF_OutData	51
Table 8: PF_OutData	51
PF_OutFlags	52
Table 9: PF_OutFlags	52
PF_OutFlags2	56
Table 10: PF_OutFlags2	56
Parameters	58
Table 11: Parameter Types	59
Slider range issues?	61
Point parameter origin	61
PF_ParamDef	61
Param zero	61
Table 12: PF_ParamDef	62
Parameter UI Flags	63
Table 13: Parameter UI Flags	63
Parameter Flags	64
Table 14: Parameter Flags	64
PF_ValueDisplayFlags	65
PF_EffectWorld / PF_LayerDef	65
Table 15: PF_EffectWorld structure	65
What happened to PF_Worlds?	66
Rowbytes in PF_EffectWorlds	67
Byte alignment	67
Deeper color	67

Accessor macros for opaque (data type) pixels	67
Table 16: PF_PixelPtr accessor macros	68
Errors	69
Table 17: Error Codes	69
Error reporting policy	69
Dig in!	70

CHAPTER 3: EFFECT DETAILS 71

Free code == GOOD	71
Accessing the After Effects function suites	71
Versioning	72
Threading	72
Memory allocation	72
Table 18: PF_HandleSuite1	73
Image buffer management functions	73
Table 19: PF_WorldSuite1	73
Same meaning, different flags	74
Iteration suites	74
Table 20: Iteration suite functions	75
Graphics utility suites	77
Transform Worlds	78
Table 21: PF_WorldTransformSuite1	78
Kernel Flags	80
Table 22: Kernel Flags	81
Fill 'em up!	82
Table 23: PF_FillMatteSuite2	82
Sampling images	83
Table 24: PF_SamplingSuite Functions (multiple suites)	83
Table 25: PF_BatchSamplingSuite1	84
Do the math for me	85
Table 26: PF_ANSICallbacksSuite1	85

Interaction callback functions	87
Table 27: Interaction Callbacks	87
Parameter checkout vs. param zero	89
Parameter checkout behavior	90
Parameter checkout and re-entrancy	90
Progress during iteration	90
Pixel aspect ratio	91
Don't assume pixels are square, or 1-to-1	92
Suggested approach	92
Applying user input in pixels	93
Test test test!	93
Parameter supervision	94
Updating parameter UI	94
Beauty is only UI deep	94
When you can change parameter values	94
Parameter Utility Suite	95
Table 28: PF_ParamUtilsSuite1	95
Global, sequence, and frame data	98
Persistence	98
Validating Sequence Data	98
Flattened and unflattened sequence data	99
Resizing sequence data	99
Arbitrary data parameters	100
Table 29: Arbitrary data selectors	100
Implementing arbitrary data	101
Arbitrary data? Re-entrancy.	102
When NOT to access arbitrary parameters	102
Changes during dialogs	103
Useful utility functions	103
PF_EffectUISuite	103
Table 30: PF_EffectUISuite	103

PF_AppSuite	103
Table 31: PF_AppSuite4	104
Advanced AppSuite: you can <i>do</i> that?!	106
Table 32: AE_AdvAppSuite2	106
Formatting time	107
Table 33: PF_AdvTimeSuite1	107
Affecting the timeline	108
Table 34: PF_AdvItemSuite1	108
Accessing auxiliary channel data	109
Table 35: PF_ChannelSuite1	109
Color parameters and floating point color	111
Table 36: PF_ColorParamSuite1	111
Motion blur	111
Working with paths	112
Accessing path data	112
Manipulating path data	112
Vertices	112
Table 37: PF_PathVertex	112
PF_PathDataSuite	113
Table 38: PF_PathDataSuite1	113
PF_PathQuerySuite	115
Table 39: PF_PathQuerySuite	115
Accessing camera and light information	116
Color space conversion	117
Table 40: Pixel Types for different color spaces	117
Table 41: color space conversion callbacks	117
Changing parameter orders, the nice way	118
Change defaults? Change IDs	119
Tips and tricks	119
Best practices	119
Responsiveness	119

Make your effect easy to find	120
Sampling pixels at (x,y)	120
Where's the center of a pixel?	120
Clean slate	121
Caching behavior	121
Some thoughts on time from a long-time developer	121
Rate x Time == PAIN	123
Testing	124

CHAPTER 4: SMARTFX125

What's new in CS3 (8.0)?	125
The way things were	125
The way things are now	125
Content Bounds	126
How to Smartify	126
PF_Cmd_SMART_PRE_RENDER	126
Table 42: PF_PreRenderExtra	127
Table 43: PF_PreRenderOutput	129
preserve_rgb_of_zero_alpha	130
rectangles	130
The "size" of a layer	131
Flag on the play	132
PF_Cmd_SMART_RENDER	132
Table 44: PF_SmartRenderExtra	132
When to access layer parameters	133
Wait, Gimme That Layer Back!	134

CHAPTER 5: EFFECT UI & EVENTS135

Table 45: Events	135
------------------------	-----

PF_EventExtra	136
Table 46: PF_EventExtra	136
PF_Context	138
Table 47: PF_Context	138
Table 48: PF_EffectWindowInfo	138
PF_EventUnion	139
Click	139
Table 49: PF_DoClickEventInfo	139
Draw	139
Table 50: PF_DrawEventInfo	139
Keydown	140
Table 51: PF_KeyDownEvent	140
AdjustCursor	142
Table 52: PF_AdjustCursorEventInfo	142
Arbitrary Parameters Event	142
Table 53: PF_ArbParamsExtra	142
Custom UI and Drawbot	143
Make Your Custom UI Look Not So “Custom”	143
Redrawing	144
PF_EffectCustomUISuite	144
Table 54: PF_EffectCustomUISuite1	144
DRAWBOT_DrawbotSuite	144
Table 55: DRAWBOT_DrawbotSuite1	144
DRAWBOT_SupplierSuite	145
Table 56: DRAWBOT_SupplierSuite1	145
DRAWBOT_SurfaceSuite	147
Table 57: DRAWBOT_SurfaceSuite1	147
DRAWBOT_PathSuite	150
Table 58: DRAWBOT_PathSuite1	150
PF_EffectCustomUIOverlayThemeSuite	151
Table 59: PF_EffectCustomUIOverlayThemeSuite1	151

The Way Custom UI Used to Be	152
What this means to you	153
How expensive?	153
UI callbacks	153
Table 60: UI Callbacks	154
Tips and tricks	156
UI Performance	156
No more black	156
How deep are my pixels?	156
Arbitrary data	156
Custom UI implementation for color sampling, using keyframes	156

CHAPTER 6: AUDIO157

Global Outflags	157
Audio Data Structures	157
Table 61: Audio data structures	157
Audio-specific Float Slider Variables	158
Flags	158
Phase	158
Curve Tolerance	158
What's zero, really?	158
Accessing Audio Data	159
Extending Audio Clips	159
Audio Considerations	159

CHAPTER 7: AEGPS160

What's new?	160
What's New in CS5?	160
What's New in CS4 (9.0)?	161
What's new in CS3 (8.0)?	161

Overview	161
AEGP communication with After Effects	162
Different tasks, same API	162
Data Types	162
Table 62: AEGP API Data Types	162
Implementation	164
Entry Point	164
The Hook-Up	165
Specialization	165
Private Data	165
Example: adding a menu item	166
Fail gracefully	166
Nasty, brutish, and short	166
Threading	167
Accessing stream values	167
Okay, what did I just get?	167
Different stream types require different accessors	168
Layers	168
Masks	168
Effects	168
Dynamic Streams	168
Pixels	168
Sounds	169
Were you just going to <i>leave</i> that data <i>lying around</i> ?	169
Table 63: Data types requiring disposal	169
AEGP Suites	170
Table 64: AEGP Suites	170
File Import Manager suite	172
Table 65: AEGP_FIMSuite3	172
Manage Projects	173
Table 66: AEGP_ProjSuite5	173

Control Items within projects	175
Table 67: AEGP_TimeDisplay2	175
Table 68: AEGP_ItemSuite8	176
Blending tables	181
Table 69: AEGP_ColorSettingsSuite2	181
Manipulate Compositions	183
Table 70: AEGP_CompSuite7	183
Work With Footage	188
Table 71: AEGP_FootageSuite5	188
Table 72: AEGP_FootageInterp structure	194
Work With Audio	196
Table 73: AEGP_SoundDataSuite1	196
Manage Layers	197
Table 74: AEGP_LayerSuite7	197
A note about layer offsets	205
Working with streams	205
Stream Suite	205
Table 75: AEGP_StreamSuite4	205
Marker Streams	212
Table 76: AEGP_MarkerSuite2	212
Dynamic Streams	214
Table 77: AEGP_DynamicStreamSuite4	215
Working with keyframes	222
Table 78: AEGP_KeyframeSuite3	222
Adding multiple keyframes	227
Working with text layers	227
Table 79: AEGP_TextDocumentSuite1	227
Working with text outlines	228
Table 80: AEGP_TextLayerSuite1	228
Render Suite	229
Table 81: AEGP_RenderSuite2	229

Communication with a layer's effects	231
Table 82: AEGP_EffectSuite3	231
Exploiting effect UI behavior to look cool	235
StreamRefs and EffectRefs	235
Registering with After Effects	235
Table 83: AEGP_RegisterSuite5	235
Managing menu items	237
Table 84: AEGP_CommandSuite1	237
Utility functions	239
Table 85: AEGP_UtilitySuite5	239
Handling Handles	243
Table 86: AEGP_MemorySuite1	244
Mask Management	245
Table 87: AEGP_MaskSuite5	245
Mask Outlines	248
Table 88: AEGP_MaskOutlineSuite2	248
Persistent Data Suite	249
Table 89: AEGP_PersistentDataSuite3	250
Working with Effects	253
Table 90: AEGP_PFIInterfaceSuite1	254
AEGP_GetEffectCameraMatrix notes	255
Render Queue Suite	255
Table 91: AEGP_RenderQueueSuite1	255
Render Queue Item Suite	256
Table 92: AEGP_RQItemSuite3	256
Render Options Suite	258
Table 93: AEGP_RenderOptionsSuite3	258
Output Module Suite	261
Table 94: AEGP_OutputModuleSuite4	262
Managing selections	266
Table 95: AEGP_CollectionSuite2	266

Ownership of collection items	267
The AEGP_World as we know it	267
Table 96: AEGP_WorldSuite3	267
Do this many times	270
Table 97: AEGP_IterateSuite1	270
Cheating: effect usage of AEGP suites	270
Depending on AEGP Queries	271
AEGP Details	271
Have a cookie	271
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() for an example.	271
Modifying items in the render queue	271
Names and Solids	271
Layer creation notes	272
Reporting errors and problems	272
Transforms: what happens first?	272
Tracked Memory	272
Accessing pixels from effect layer parameters	272

CHAPTER 8: ARTISANS273

Interactive Artisans	273
What's new in 7.0?	273
Artisan Data Types	274
Table 98: Data types used in the Artisan API	274
Horz? Vert?	274
Implementation and Design	274
3D compositing, not modeling	274
Registering an Artisan	275

Table 99: Artisan Entry Points	275
Render Suite	278
Table 100: AEGP_RenderSuite	278
The World Is Your Canvas	279
Table 101: AEGP_CanvasSuite7	279
Convert between different contexts	288
Table 102: AEGP_ArtisanUtilSuite1	288
Track Mattes and Transform functions	289
Table 103: AEGP_CompositeSuite2	289
Smile! Cameras	292
Table 104: AEGP_CameraSuite2	292
Notes regarding camera behavior	293
Orthographic camera matrix	293
Focus on Focal	293
Film size	293
Hit the lights!	294
Table 105: AEGP_LightSuite2	294
Notes on light behavior	294
How should I draw that?	294
Transform Conventions	295
Query transform functions	295
Table 106: AEGP_QueryXformSuite2	295
Interactive drawing functions	298
Table 107: PR_InteractiveDrawProcs	298
Notes on Query Time functions	298

CHAPTER 9: AEIOS299

What's New in CS5?	299
What's New in CS3 (8.0)?	299
What can AEIOs do?	299
What they cannot do	300

AEIO, or AEGP?	300
How they do it	300
What would After Effects do?	300
Registering your AEIO	300
AEIO_ModuleInfo	301
Table 108: AEIO_ModuleInfo	301
Behavior Flags	302
Table 109: AEIO_ModuleFlags	302
AEIO_ModuleFlags2	304
Table 110: AEIO_ModuleFlags2	304
New Kids on the Function Block	304
Table 111: AEIO_FunctionBlock4	305
What Goes In	316
Table 112: AEGPIOInSuite4	316
What Goes Out	320
Table 113: AEGPIOOutSuite4	320
User Data vs. Options	325
Calling sequence	325

CHAPTER 10: PREMIERE PRO327

Different pixel formats	327
32-bit floating point support	328
Multithreading	328
Parameters	328
Time differences	328
Plug-ins... Reloaded	329
Effect thumbnail previews	330
General guidelines	331
But...why'd you LOAD it, if you can't RUN it?!	331
Other Hosts?	332
Reality Sandwich	332

Function and Suite Reference	333
General Reference	336

1 • INTRODUCTION

Welcome to the Adobe® After Effects® CS5 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/>

If you have questions about the APIs described in this document, or about integration with After Effects, your question may already be answered on the After Effects SDK forum at: http://forums.adobe.com/community/aftereffects_general_discussion/aftereffects_sdk

SDK AUDIENCE

You must be a proficient C or 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 After Effects Classroom in a Book*](#), 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 3.1 for Mac OS 10.5.7, or XCode 3.2 on Mac OS 10.6, and Microsoft Visual Studio .NET 2008 (version 9.0) SP 1 for Windows Vista 64 or 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.

WHAT PLUG-INS ARE

Effect plug-ins process video and/or audio data. Their parameters can vary over time, and they may access other layers and parameters at different times. They may make use of After Effects' built-in parameters, or may provide custom UI for the parameter controls.

After Effects General Plug-ins (AEGPs) can manipulate nearly every element of After Effects projects (from keyframes to footage paths) and preferences. They can also 'hook' (respond to) and trigger After Effects' internal commands. They can even call scripts. Most types of AEGPs are discussed separately in the [AEGP](#) chapter. [AEIO](#) plug-ins add support for new and different file types, and support audio (unlike the old Photoshop format plug-in API). [Artisans](#) render the entire composition instead of a single layer. While After Effects still supports Photoshop format plug-ins and filters, as well as Foreign Project Format (FPF) plug-ins, these APIs have been deprecated in favor of the expanded AEGP API.

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, or [PiPL](#) resource, on both platforms. They are compiled using the SDK header files, which expose various After Effects functionality to be used by the plug-in.

WHERE 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). [AEIOs](#) and Photoshop Format plug-ins 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. Format plug-ins also appear as available output formats in the render queue. Artisans appear in the *Rendering Plug-in* drop-down in the *Advanced* tab of the *Composition Settings* dialog.

AEGPs can add items to any After Effects menu, and are indistinguishable from After Effects' own menu items.

HOW AFTER EFFECTS INTERACTS WITH PLUG-INS

After Effects sends command selectors (and relevant information) to the effect plug-in's entry point function designated in the effects' [PiPL](#) 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.

SDK CONTENTS

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

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. Scripting can even achieve some UI integration with custom panels. 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. You may download the After Effects Scripting Guide from the Adobe Developer Connection website at: <http://www.adobe.com/devnet/aftereffects/>

PIXEL BENDER FILTERS

Pixel Bender filters are a good choice for non-commercial effects where processing of any pixel has minimum dependence on the values of other pixels. Pixel Bender filters are much easier to write than effects plug-ins, are parallel-processed, support all bit-depths, and are currently supported starting in After Effects CS4, Photoshop CS4, and Flash Player 10.

Pixel Bender is best suited for the kind of algorithms where processing of any pixel has minimum dependence on the values of other pixels. For example, you can write a kernel that efficiently manipulates image brightness, because the brightness of each pixel can be changed independently. On the other hand, you couldn't use Pixel Bender to compute a histogram, because a histogram needs values of all the pixels.

For those considering commercial development of Pixel Bender filters, a couple caveats: As of CS4, there is no encryption and watermarking support to protect code and provide trial versions of filters. There is also currently no way for an AE plug-in to call Pixel Bender filters to do video processing. Stay tuned for further developments!

The latest Pixel Bender Toolkit, the Pixel Bender Exchange (a place to filters with others), and the Pixel Bender forum are at: <http://labs.adobe.com/technologies/pixelbender/>.

PREMIERE PRO IMPORTERS

Premiere Pro importers provide support for importing media into applications across most applications in the Adobe Creative Suite Production Premium, including Premiere Pro, After Effects, Encore, and Soundbooth. Because of this broader compatibility, we recommend developing a Premiere Pro importer, rather than developing an AEIO using this SDK. The Premiere Pro SDK is available at: <http://www.adobe.com/devnet/premiere/>

HARDWARE OUTPUT APIS

QuickTime VOut components are supported on both Mac OS and Windows for video output to hardware. They appear in After Effects in Edit>Preferences>Video Preview>Output Device. Note that After Effects also has it's own simple API for this, as demonstrated in the EMP sample project.

WHAT'S NEW?

WHAT'S NEW IN CS5 (10.0)?

The biggest change is 64-bit support. As a result of the 64-bit port, we have had to change code that assumed a long was 32-bits. 32-bit binaries are not supported in CS5. The sample projects have been ported to 64-bit on Windows and Mac OS.

Custom UI drawing for effects has also changed. The new [Drawbot](#) suites replace the previous drawing methods. See the ColorGrid sample project for an example of how to use it.

Multibyte-encoded platform paths have been upgraded to Unicode. This required changes to the [AEGP entry point function](#), various AEGP suites, and the [AEIO_FunctionBlock](#).

We have made type changes to allow larger frame sizes in the future. PF_Rect, PF_Point, and A_Rect now use A_long components, and the old versions with A_short components have been renamed to PF_LegacyRect, PF_LegacyPoint, and A_LegacyRect.

Structure packing is now explicitly set to 8 bytes. Previously, all A_type structures were aligned based only on the project settings for the plug-in. The alignment was never explicitly locked down. If any 3rd party is serializing (and writing to disk) an A_type structure, they will need to lock down the pragma pack to what it was before CS5, and write a conversion routine that packs the old data into the new 8-byte aligned structures.

The new flag [PF_OutFlag2_AUTOMATIC_WIDE_TIME_INPUT](#) provides smarter caching logic than PF_OutFlag_WIDE_TIME_INPUT. In addition, if caching any time-dependent data in sequence data (or anywhere else), you must [validate that cache](#) using the new call [PF_HaveInputsChangedOverTimeSpan](#) before using the time-dependent data.

We have phased out the old PF_Suite_Helper utilities in favor of the AEFX_SuiteHelper utilities, which we are using internally. Any code that used these old utilities should pick up the new ones.

DO I HAVE TO UPDATE MY PLUG-INS FOR CS5?

Yes, CS5 does not support the 32-bit plug-ins developed with the CS4 or earlier SDKs. Fortunately, we've kept other API changes to a minimum, so you can focus on the 64-bit port.

QUICK TIPS FOR 64-BIT WINDOWS PORTING

- 1) Open your Visual Studio solution file in Visual Studio 2008. Perform the automatic conversion to upgrade the solution from a previous version.
- 2) Add a new solution platform of type "x64".
- 3) In the project's Configuration Properties > Linker > Advanced, set Target Machine to "MachineX64".
- 4) In your project's PiPL (.r file), add the 64-bit entry point by replacing:

```
#ifdef AE_OS_WIN
    CodeWin32X86 {"EntryPointFunc"},
#else
```

with:

```
#ifndef AE_OS_WIN
    #ifdef AE_PROC_INTELx64
        CodeWin64X86 { "EntryPointFunc" },
    #else
        CodeWin32X86 { "EntryPointFunc" },
    #endif
#else
```

5) Also in the PiPL, if the `AE_Effect_Spec_Version` isn't set to `PF_PLUG_IN_VERSION` and `PF_PLUG_IN_SUBVERS`, defined in `AE_EffectVers.h`, then make it so (see any effect's PiPL for an example). If an effect's PiPL specifies an `AE_Effect_Spec_Version` earlier than the new version, 13.0, the host will mark the effect as invalid.

6a) If your plug-in is an effect that uses custom UI, update it to use [Drawbot](#).

6b) If your plug-in is an AEGP, update your [AEGP entry point](#), any AEGP suites, and your `AEIO_FunctionBlock`.

7) Now you'll need to port any code that assumed 32-bit compilation: Update the `refcons` used in `iterate` calls from `long*` to `void*`. Update suite usage for suites that are no longer supported. And so on.

Voila! The plug-in project is ready to compile a 64-bit binary!

QUICK TIPS FOR 64-BIT MAC OS PORTING

1) Open your XCode project file. In the project's info panel, in the General tab, set "Base SDK:" to "Mac OS 10.5".

2) In the Build tab, set the Architectures setting to build a 64-bit binary. Note that if you have already set Target-specific settings in the XCode project, this Architecture setting will not take effect, and you will need to open the target's info panel and set it there. You can confirm that the plug-in is being built for the correct architecture from the Terminal, by running "`otool -f <filename>`", where the filename is the binary inside the Contents/MacOS/ folder of the plug-in bundle.

3) For anything that includes Cocoa, you will need to set "Compile Sources As" to "Objective-C++".

4) In your project's PiPL (.r file), add the 64-bit entry point by finding the line:

```
CodeMacIntel32 { "EntryPointFunc" },
```

and adding the following line below:

```
CodeMacIntel64 { "EntryPointFunc" },
```

5) Also in the PiPL, if the `AE_Effect_Spec_Version` isn't set to `PF_PLUG_IN_VERSION` and `PF_PLUG_IN_SUBVERS`, defined in `AE_EffectVers.h`, then make it so (see any effect's PiPL for an example). If an effect's PiPL specifies an `AE_Effect_Spec_Version` earlier than the new version, 13.0, the host will mark the effect as invalid.

6a) If your plug-in is an effect that uses custom UI, update it to use [Drawbot](#).

6b) If your plug-in is an AEGP, update your [AEGP entry point](#), any AEGP suites, and your `AEIO_FunctionBlock`.

7) Now you'll need to port any code that assumed 32-bit compilation: Update the `refcons` used in `iterate` calls from `long*` to `void*`. Update suite usage for suites that are no longer supported. And so on.

Voila! The plug-in project is ready to compile a 64-bit binary!

WHAT ABOUT BACKWARDS COMPATIBILITY?

Use the CS5 headers for creating a 64-bit binary. These headers cannot be used for creating plug-ins for older hosts, because there have been significant ABI changes to the API. Use the CS4 headers for creating a 32-bit binary.

Note that CS5 requires a PiPL set to version 13.0, whereas older hosts require a PiPL of 12.x.

On Mac OS, the 32-bit and 64-bit plug-ins can be combined into a fat binary, using the Mac OS tool 'lipo'.

WHAT WAS NEW IN CS4 (9.0)?

Very little has been added to the AE API. We have published new versions of many suites to support Unicode strings of infinite length. A bonus for non-Unicode plug-ins: all multi-byte strings that still exist related to layer names are now up to 255 chars (up to 128 double-byte chars).

[AEGP_DynamicStreamSuite](#) has been updated to support the new Position keyframes that have been separated into individual XYZ properties. [AEGP_MarkerSuite](#) has been updated to support marker durations.

Apart from the AE API, we have added initial support for Adobe's new Pixel Bender technology. [Pixel Bender filters](#) are easy to write, are parallel-processed, support all bit-depths, and are currently supported in AE CS4, Photoshop CS4, and Flash Player 10.

WHAT WAS NEW IN CS3 (8.0)?

We've kept the API changes to a minimum this time. SmartFX (required for floating point color support in effects) has been a substantial undertaking for developers, and we understand the last thing you need right now is another large API change.

Instead, we've done some housekeeping. We've simplified multi-platform development with `AEConfig.h`, the same definitions we use internally. `AE_GeneralPlugPre.h` and `AE_GeneralPlugPost.h` ensure definition and structure alignment consistent with After Effects (departure from such consistency having caused some pernicious bugs in previous releases).

After Effects now 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 SDK](#). 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.

...AND WHAT WAS NEW BEFORE CS3?

For history beyond the previous version, see obsolete copies of the SDK (which we don't provide; if someone wants you to develop for antique software, they'd best provide the SDK).

WHY DEVELOP AFTER EFFECTS PLUG-INS?

Easily implement pixel processing algorithms, keyframe manipulators, and project management tools as After Effects plug-ins. Effects are a great way to explore pixel manipulation without writing loads of infrastructure, or marrying yourself to one platform's imaging architecture. Take advantage of the compositing and parameter management After Effects provides. Also, writing plug-ins can actually be fun (more fun than writing another PHP/.NET/SQL/AJAX client, anyway...yawn).

Our evangelical pitch aside, the After Effects team does everything it can to make our product accessible, extensible, and a good workflow citizen. The APIs reflect these goals.

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.](#)

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. Starting in CS4, the GLator sample demonstrates OpenGL usage for video rendering.

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.

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 instead.
Grabba	Gets frames (formatted as the plug-in requests) from any composition in the project.

TABLE 1: SAMPLE PROJECT DESCRIPTIONS

Project	Description
IO	Exercises the After Effects Input/Output (AEIO) API. Supports the fictitious <code>.fak</code> 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 instead.
Mangler	Mangler is a keyframer demonstrating the use of an ADM palette, just like our own.
Panelator	New in CS3. Creates a panel that can be docked along with the rest of the standard panels.
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) <code>.sdk</code> 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.
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. We have left this plug-in remain in C code, for nostalgia's sake.
Convolutrix	Exercises our image convolution callbacks.
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.
GLator	New in CS4. A sample effect that uses OpenGL to resize the video and create a reflection beneath it.
Paramarama	Exercises wayward param types not used in other sample.
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.

TABLE 1: SAMPLE PROJECT DESCRIPTIONS

Project	Description
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.
Shifter	Shifts an image in the output buffer, and exercises our 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 ECP UI	Implements a very boring custom user interface in the effect controls window, and shows how to respond to numerous UI events.
Supervisor	Shows how to control parameters (both values and UI) based on the value of other parameters. Premiere Pro compatible.
GPs	
EMP	External Monitor Preview. Use this as a starting point for adding support to output video from the composition panel to video hardware.

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 plug-ins; 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 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. We've tried to make setting your debug output directory simple. On Windows, change the value of the `AE_PLUGIN_BUILD_DIR` environment variable as needed. On Mac OS X, use XCode's preferences to specify the output directory for all projects. Between this configurability, projects which stress your plug-in, and tools provided your development environment, you're well on your way to shipping some tested code.

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.

Change the output directory of your project to build into After Effects' plug-in directory. Next, 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. Specify the Executable Path to the After Effects executable file.

SLICK DEBUGGING TRICKS

You need not quit After Effects each time to reload your plug-in after rebuilding it. On Mac OS, use `Ctrl-Clear` (from the numeric keypad) to purge your plug-in; `Ctrl-Alt-/` (keypad slash) on Windows. *Edit > Purge > All* has the same effect (and clears all buffers). Only plug-ins that were present when After Effects was started will be available; to *add* other plug-ins, you must restart the application. This trick will not work on plug-ins which have set the [super secret PiPL bit](#).

WHERE PLUG-INS ARE INSTALLED

In CS3 and later, the applications look for plug-ins in a new common location. Installing your plug-ins here 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
```

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.

COMPATIBILITY ACROSS MULTIPLE 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>version`. If you don't update your project's PiPL to request new versions, new API features will not be available to your plug-in.

TABLE 2: EFFECT API VERSIONS

Release	Version
CS5 (10.0)	13.0
CS4 (9.0)	12.14
CS3 (8.0)	12.13

TABLE 2: EFFECT API VERSIONS

Release	Version
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](#)>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](#).

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](#) 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, 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.

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.

LOCALIZATION

Thanks to the new Unicode support in CS4, localization is much less painful. For non-Unicode strings, such as in older function suites, we expect strings to be multi-byte encoded using the application's current locale. On Windows, you can use the [`WideCharToMultiByte\(\)`](#) function, specifying `CP_OEMCP` as the first argument. On Mac OS, use the encoding returned by `GetApplicationTextEncoding()`.

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

For all effect plug-ins, the entry point function (the name of which is specified in the PiPL) 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)
```

Before each call to an effect's entry point, After Effects updates [PF_InData](#) and the plug-in's parameter array (except as noted). After the plug-in returns from its call, After Effects checks [PF_OutData](#) for changes and, when appropriate, uses the output the effect has rendered.

TABLE 3: EFFECT PLUG-IN ENTRY POINT

Argument	Purpose
cmd	After Effects uses the command selector to tell the plug-in what to do.
in_data	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.
out_data	Pass back information to After Effects by setting fields within <code>out_data</code> .

TABLE 3: EFFECT PLUG-IN ENTRY POINT

Argument	Purpose
params	An array of the plug-in's parameters at the time provided in <code>in_data>current_time</code> . <code>params[0]</code> is the input image (a PF_EffectWorld) to which the effect should be applied. These values are only valid during certain selectors (this is noted in the selector descriptions). Parameters are discussed at length here .
output	The output image, to be rendered by the effect plug-in and passed back to After Effects. Only valid during certain selectors.
<code>extra</code>	The <code>extra</code> parameter varies with the command sent or (in the case of PF_Cmd_EVENT) the event type . Used primarily for event management and parameter supervision .

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*...

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 <code>out_data>return_msg</code> 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 <code>PF_OutData</code> fields (including <code>out_data>my_version</code>) 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 . Also, register custom user interface elements. Set <code>PF_OutData>num_params</code> to match your parameter count.

TABLE 4: COMMAND SELECTORS

Selector	Response
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 is initialized at this time.
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	<p>This is only sent if the plug-in sets PF_OutFlag_SEQUENCE_DATA_NEEDS_FLATTENING. Sent when saving and when duplicating the sequence. Flatten sequence data containing pointers or handles so it can be written to disk. This will be saved with the project file. Free the unflat data and set the <code>out_data>sequence_data</code> to point to the new flattened data. Flat data must be correctly byte-ordered for file storage.</p> <p>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.</p>
PF_Cmd_SEQUENCE_SETDOWN	Free all sequence data.
Frame Selectors Passed for each frame (or audio blip) to be rendered by your plug-in.	

TABLE 4: COMMAND SELECTORS

Selector	Response
PF_Cmd_FRAME_SETUP	<p>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.</p> <p>If you set <code>width</code> and <code>height</code> to 0, After Effects ignores your response to the following <code>PF_Cmd_RENDER</code>.</p> <p>NOTE: If <code>PF_Outflag_I_EXPAND_BUFFER</code> is set, you will receive this selector (and <code>PF_Cmd_FRAME_SETDOWN</code>) twice, once without <code>PF_Cmd_RENDER</code> between them. This is so we know whether or not the given layer will be visible.</p> <p>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 <code>PF_Cmd_RENDER</code>.</p>
PF_Cmd_RENDER	<p>Populate <code>output</code> with effect-ed data. All fields in <code>PF_InData</code> are valid. If your response to this selector is interrupted (your calls to <code>PF_ABORT</code> or <code>PF_PROGRESS</code> returns an error code), your results will not be used. You cannot delete <code>frame_data</code> during this selector; you must wait until <code>PF_Cmd_FRAME_SETDOWN</code>.</p>
PF_Cmd_FRAME_SETDOWN	<p>Free any frame data allocated during <code>PF_Cmd_FRAME_SETUP</code>.</p>
PF_Cmd_AUDIO_SETUP	<p>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 <code>startsample</code> and <code>endsample</code> field in <code>PF_OutData</code>.</p>
PF_Cmd_AUDIO_RENDER	<p>Populate <code>PF_OutData.dest_snd</code> with effect-ed audio. All fields in <code>PF_InData</code> are valid. If your response to this selector is interrupted (your calls to <code>PF_ABORT</code> or <code>PF_PROGRESS</code> returns an error code), your results will not be used.</p>
PF_Cmd_AUDIO_SETDOWN	<p>Free memory allocated during <code>PF_Cmd_AUDIO_SETUP</code>.</p>
PF_Cmd_SMART_PRE_RENDER	<p>SmartFX only. Identify the area(s) of input the effect will need to produce its output, based on whatever criteria the effect implements.</p>
PF_Cmd_SMART_RENDER	<p>SmartFX only. Perform rendering and provide output for the area(s) the effect was asked to render.</p>

TABLE 4: COMMAND SELECTORS

Selector	Response
<p>Messaging</p> <p>The communication channel between After Effects and your plug-in.</p>	
PF_Cmd_EVENT	<p>This selector makes use of the <code>extra</code> parameter; the type of event to be handled is indicated by the <code>e_type</code> field, a member of the structure pointed to by <code>extra</code>. See Effect UI & Events.</p>
PF_Cmd_USER_CHANGED_PARAM	<p>The user changed a parameter value. You will receive this command only if you've set the PF_ParamFlag_SUPERVISE flag. You modify the parameter to control values, or make one parameter's value affect others. A parameter can be modified by different actions.</p>
PF_Cmd_UPDATE_PARAMS_UI	<p>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().</p> <p>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.</p> <p>This command will only be sent regularly if PF_OutFlag_SEND_UPDATE_PARAMS_UI was set in the PiPL, and during PF_Cmd GLOBAL SETUP.</p> <p>NOTE: Never check out parameters during this selector. Recursive badness is almost guaranteed to result.</p>
PF_Cmd_DO_DIALOG	<p>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 <code>params</code> 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.</p>
PF_Cmd_ARBITRARY_CALLBACK	<p>Manage your arbitrary data type. You'll only receive this if you've registered a custom data type parameter. The <code>extra</code> parameter indicates which handler function is being called. Custom data types are discussed further in Implementation.</p>

TABLE 4: COMMAND SELECTORS

Selector	Response
PF_Cmd_GET_EXTERNAL_DEPENDENCIES	<p>Only sent if PF_OutFlag_I_HAVE_EXTERNAL_DEPENDENCIES was set during <i>PF Cmd GLOBAL SETUP</i>.</p> <p>Populate a string handle (in the <code>PF_ExtDependenciesExtra</code> pointed to by <code>extra</code>) with a description of your plug-in's dependencies, making sure to allocate space for the terminating NULL character. Return just a null character (<code>'\0'</code>) if there are no dependencies to report.</p> <p>If the check type is <code>PF_DepCheckType_ALL_DEPENDENCIES</code>, 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 <code>PF_DepCheckType_MISSING_DEPENDENCIES</code>.</p>
PF_Cmd_COMPLETELY_GENERAL	<p>Respond to an AEGP. The <code>extra</code> parameter points to whatever parameter the AEGP sent. AEGPs can only communicate with effects which respond to this selector.</p>
PF_Cmd_QUERY_DYNAMIC_FLAGS	<p>New in 5.0. Sent only to plug-ins which have specified <code>PF_OutFlag2_SUPPORTS_QUERY_DYNAMIC_FLAGS</code> in <code>PF_OutFlags2</code>, in their PiPL and during <i>PF Cmd GLOBAL SETUP</i>. This selector will be sent at arbitrary times. In response, the effect accesses its (non-layer) parameters, and determines whether or not the following <code>outFlags</code> and <code>outFlag2s</code> need to be set</p> <p>PF_OutFlag_WIDE_TIME_INPUT PF_OutFlag_NON_PARAM_VARY PF_OutFlag_I_USE_SHUTTER_ANGLE PF_OutFlag2_I_USE_3D_CAMERA PF_OutFlag2_I_USE_3D_LIGHTS PF_OutFlag2_REVEALS_ZERO_ALPHA</p> <p>After Effects uses this information for caching and optimization purposes, so try to respond as quickly as possible.</p>

RESPONDING TO SELECTORS

With each command selector, effects receive information from After Effects in [PF_InData](#), input and parameter values in `params` (an array of parameter descriptions including the input layer). They send information back to After Effects in [PF_OutData](#), and (when appropriate) render output to a `PF_LayerDef`, also called a [PF_EffectWorld](#). During events, they receive event-specific information in `extra`.

All the information provided by After Effects, combined with the provided callbacks and function suites, combine to make the effects API very powerful and flexible.

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](#), then [PF_Cmd_PARAM_SETUP](#). Each time the user adds the effect to a layer, [PF_Cmd_SEQUENCE_SETUP](#) is sent.

For each frame rendered by a non-SmartFX effect, After Effects sends [PF_Cmd_FRAME_SETUP](#), then [PF_Cmd_RENDER](#), then [PF_Cmd_FRAME_SETDOWN](#). All effect plug-ins must respond to [PF_Cmd_RENDER](#). For SmartFX, [PF_Cmd_SMART_PRE_RENDER](#) may be sent any number of times, before a single [PF_Cmd_SMART_RENDER](#) is sent.

[PF_Cmd_SEQUENCE_SETDOWN](#) is sent on exit, when the user removes an effect or closes the project. [PF_Cmd_SEQUENCE_RESETUP](#) is sent when a project is loaded or when the layer to which it's applied changes. [PF_Cmd_SEQUENCE_FLATTEN](#) is sent when the After Effects project is written out to disk, but only if [PF_OutFlag_SEQUENCE_DATA_NEEDS_FLATTENING](#) was set during [PF_Cmd_GLOBAL_SETUP](#).

[PF_Cmd_ABOUT](#) is sent when the user chooses *About...* from the effect controls palette (ECP).

[PF_Cmd_GLOBAL_SETDOWN](#) 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.

WHAT'S THE DIFFERENCE?

There is a subtle difference between [PF_Cmd_USER_CHANGED_PARAM](#) and [PF_Cmd_UPDATE_PARAMS_UI](#). 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](#) 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 Interaction Callback Functions .
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'. 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.
current_time	<p>The time of the current frame being rendered, valid during PF Cmd RENDER. 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.</p> <p>The current frame number is current_time divided by time_step. The current time in seconds is current_time divided by time_scale.</p> <p>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!</p>

TABLE 5: PF_INDATA

Name	Description
time_step	<p>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 (<code>local_time_step</code>). This value can be converted to seconds by dividing by <code>time_scale</code>.</p> <p>When calculating other source frame times, such as for PF_CHECKOUT_PARAM, use this value rather than <code>local_time_step</code>.</p> <p>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.</p> <p>Can differ from <code>local_time_step</code> 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.</p> <p>This value will be 0 during PF_Cmd_SEQUENCE_SETUP if it is not constant for all frames. It will be set correctly during <code>PF_Cmd_FRAME_SETUP</code> and <code>PF_Cmd_FRAME_SETDOWN</code> selectors. WARNING: This can be zero, so check it before you divide.</p>
total_time	<p>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 <code>time_scale</code>.</p>
local_time_step	<p>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 <code>time_step</code>, this value is constant from one frame to the next. This value can be converted to seconds by dividing by <code>time_scale</code>.</p> <p>For a step value that is constant over the entire frame range of the layer, use <code>local_time_step</code>, which is based on the composition's framerate and layer stretch.</p>
time_scale	<p>The units per second that <code>current_time</code>, <code>time_step</code>, <code>local_time_step</code> and <code>total_time</code> are in. If <code>time_scale</code> is 30, then the units of <code>current_time</code>, <code>time_step</code>, <code>local_time_step</code> and <code>total_time</code> are in 30ths of a second. The <code>time_step</code> might then be 3, indicating that the sequence is actually being rendered at 10 frames per second. <code>total_time</code> might be 105, indicating that the sequence is 3.5 seconds long.</p>
field	<p>Valid only if PF_OutFlag_PIX_INDEPENDENT was set during PF_Cmd_GLOBAL_SETUP. Check this field to see if you can process just the upper or lower field.</p>

TABLE 5: PF_INDATA

Name	Description
shutter_angle	<p>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. <code>shutter_angle == 180</code> means the time interval between <code>current_time</code> and <code>current_time + 1/2 time_step</code>. Valid only if PF_OutFlag_I_USE_SHUTTER_ANGLE was set during PF_Cmd GLOBAL SETUP.</p> <p>See the section on Motion Blur for details on how to implement motion blur in your effect.</p>
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 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 PF_Cmd SEQUENCE SETUP , PF_Cmd SEQUENCE RESETUP , PF_Cmd FRAME SETUP and PF_Cmd FRAME RENDER .
downsample_y	
pixel_aspect_ratio	Pixel aspect ratio (width over height).
in_flags	Unused.
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 <code>total_time</code> expressed in samples.
src_snd	PF_SoundWorld describing the input sound. Audio-specific.

TABLE 5: PF_INDATA

Name	Description
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](#).)

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` by setting `PF_OutFlag_USE_OUTPUT_EXTENT` in `PF_OutData` during `PF_Cmd_GLOBAL_SETUP` (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`. To get the output extent in the coordinate system of the output buffer, offset the `extent_hint` by the `PF_InData>output_origin_x` and `y` 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](#).

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 `output_origin_x/y` in [PF_InData](#) during [PF_Cmd_FRAME_SETUP](#). This shift is reported to subsequent effects in the [pre_effect_source_origin_x/y](#). 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
<code>my_version</code>	Set this flag (using the <code>PF_VERSION</code> macro) to the version of your plug-in code. After Effects uses this data to decide which of duplicate effects to load.
<code>name</code>	Copy your plug-in's name here. After Effects checks this after PF_Cmd_GLOBAL_SETUP .
<code>global_data</code>	Handle which will be returned to you in PF_InData with every call. Use After Effects' memory allocation functions.
<code>num_params</code>	After Effects checks this field against the number of calls made to PF_ADD_PARAM , as well as the implicit input layer.
<code>sequence_data</code>	Allocatable upon receiving PF_Cmd_SEQUENCE_SETUP , this handle will be passed back to you in PF_InData during all subsequent calls.
<code>flat_sdata_size</code>	Unused (After Effects knows the size, because you used its allocation functions to get the memory in the first place).
<code>frame_data</code>	Handle you (might have) allocated during PF_Cmd_FRAME_SETUP . This is never written to disk; it was used to pass information from your PF_Cmd_FRAME_SETUP response to your PF_Cmd_RENDER or PF_Cmd_FRAME_SETDOWN (which you must do if you resize the output buffer). Otherwise, this memory is rarely used.
<code>width, height, origin</code>	Set during PF_Cmd_FRAME_SETUP if the output image size differs from the input. <code>width</code> and <code>height</code> are the size of the output buffer, and <code>origin</code> is the point the input should map to in the output. To create a 5-pixel drop shadow up and left, set <code>origin</code> to (5, 5).
out_flags	Send messages to After Effects. OR together multiple values.
<code>return_msg</code>	After Effects displays any C string you put here (checked and cleared after every command selector).
<code>start_sampL</code>	Used only for audio commands
<code>dur_sampL</code>	
<code>dest_snd</code>	
out_flags2	Send messages to After Effects. OR together multiple values.

PF_OUTFLAGS

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](#). 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 , the plug-in's resources are always open, but unavailable at all other times (except during PF_Cmd_ABOUT and PF_Cmd_DO_DIALOG), unless this flag has been set. Set if you need access to resources at any time other than during PF_Cmd_GLOBAL_SETUP . 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	<p>The effect checks out a parameter at a time other than current_time. 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.</p> <p>New in CS5, if you set this flag, we strongly recommend you also set PF_OutFlag2_AUTOMATIC_WIDE_TIME_INPUT for better performance.</p>
PF_OutFlag_NON_PARAM_VARY	<p>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.</p> <p>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.</p> <p>Set during PF_Cmd_GLOBAL_SETUP. Can be overridden dynamically if needed during PF_Cmd_QUERY_DYNAMIC_FLAGS. Turn this off whenever possible to improve performance.</p>
PF_OutFlag_RESERVED6	Unused. Formerly PF_OutFlag_SEND_PARAMS_UPDATE. Replaced by PF_OutFlag_REFRESH_UI .

TABLE 7: PF_OUTFLAGS

Flag	Indicates
PF_OutFlag_SEQUENCE_DATA_NEEDS_FLATTENING	Sequence data contains referencing items (pointers, handles), which must be flattened for storage and unflattened for use. See PF Cmd SEQUENCE RESETUP .
PF_OutFlag_I_DO_DIALOG	<p>Effect displays a dialog in response to PF Cmd DO_DIALOG. Set during PF Cmd GLOBAL SETUP, checked during PF Cmd SEQUENCE SETUP.</p> <p>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.</p>
PF_OutFlag_USE_OUTPUT_EXTENT	<p>Effect honors the output extent_rect. Set during PF Cmd GLOBAL SETUP. See details at the end of the chapter for proper usage.</p> <p>Note: Obsolete for SmartFX.</p>
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 after PF Cmd SEQUENCE SETUP . Set during PF Cmd SEQUENCE RESETUP , not during PF Cmd GLOBAL SETUP .
PF_OutFlag_DISPLAY_ERROR_MESSAGE	Display the contents of return_msg 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	<p>Effect expands the output buffer. Set during PF Cmd GLOBAL SETUP. Set this flag and PF_OutFlag_USE_OUTPUT_EXTENT to use the intersection of the output extent_rect and your new buffer size during PF Cmd FRAME SETUP. Use pre_effect_source_origin fields to detect other transformations.</p> <p>Note: Only set this flag if you need to; it drastically reduces caching efficiency.</p> <p>Note: Obsolete for SmartFX.</p>

TABLE 7: PF_OUTFLAGS

Flag	Indicates
PF_OutFlag_PIX_INDEPENDENT	<p>A given pixel is independent of the pixels around it. Set during PF Cmd GLOBAL SETUP. Color correction effects are typically pixel independent, distortions are not.</p> <p>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.</p>
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 Cmd GLOBAL SETUP .
PF_OutFlag_I_SHRINK_BUFFER	<p>The effect shrinks its buffer based on the extent_rect in order to be more memory efficient. Set during PF Cmd GLOBAL SETUP whenever possible.</p> <p>Note: Obsolete for SmartFX.</p>
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 messages. Set during PF Cmd GLOBAL SETUP .
PF_OutFlag_RESERVED5	Unused.
PF_OutFlag_REFRESH_UI	Refresh the entire effect controls, composition, and layer windows. Set during PF Cmd EVENT , PF Cmd RENDER , and PF Cmd DO DIALOG . If refreshing custom UI during PF_Cmd_EVENT, we recommend using the new redraw mechanism with finer granularity.
PF_OutFlag_NOP_RENDER	Set this flag during PF Cmd FRAME SETUP to invalidate the current render.
PF_OutFlag_I_USE_SHUTTER_ANGLE	Indicates rendered images depend upon the value of shutter_angle .
PF_OutFlag_I_USE_AUDIO	Effect's parameters depend on audio data, obtained using PF_CHECKOUT_LAYER_AUDIO .
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.
PF_OutFlag_FORCE_RERENDER	<p>Effect made a change that requires a re-render.</p> <p>PF_ChangeFlag_CHANGED_VALUE also forces a re-render.</p>

TABLE 7: PF_OUTFLAGS

Flag	Indicates
PF_OutFlag_PiPL_OVERRIDES_OUTDATA_OUTFLAGS	After Effects will use PiPL outflags, and ignore those set during PF Cmd GLOBAL SETUP .
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 .
PF_OutFlag_DEEP_COLOR_AWARE	The effect handles 16-bpc color.
PF_OutFlag_SEND_UPDATE_PARAMS_UI	Set this flag during PF Cmd GLOBAL SETUP to receive PF Cmd UPDATE PARAMS UI .
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 or PF_OutFlag_AUDIO_EFFECT_ONLY .
PF_OutFlag_AUDIO_IIR	Set during PF Cmd GLOBAL SETUP 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 , 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). In response to PF Cmd AUDIO SETUP , 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 based on the last cached delay line. Use PF_HasParamChanged to determine whether or not your cache is valid.
PF_OutFlag_I_SYNTHESIZE_AUDIO	Set during PF Cmd GLOBAL SETUP time if the effect generates audio, even when passed silence. You must also set either PF_OutFlag_AUDIO_EFFECT_TOO or PF_OutFlag_AUDIO_EFFECT_ONLY .
PF_OutFlag_AUDIO_EFFECT_TOO	Set during PF Cmd GLOBAL SETUP if the effect alters audio.
PF_OutFlag_AUDIO_EFFECT_ONLY	Set during PF Cmd GLOBAL SETUP 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](#) 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 . Must be set in the PiPL and during PF Cmd GLOBAL SETUP .
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 PF Cmd PARAM SETUP , by setting the PF_ParamFlag_START_COLLAPSED flag in PF_ParamFlags . 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 .
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	Added for Premiere usage only; indicates that the plug-in can perform final composite.
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.) The origin of the trimmed buffer can be found in in_data>pre_effect_source_origin . 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.

TABLE 8: PF_OUTFLAGS2

Flag	Indicates
PF_OutFlag2_REVEALS_ZERO_ALPHA	<p>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. Note that, while this flag can cause changes to the size of the extent_hint, it will not change the image buffer size.</p> <p>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.</p> <p>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.</p>
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	<p>The effect supports 32-bpc floating point color representation.</p> <p>NOTE: PF_OutFlag2_SUPPORTS_SMART_RENDER must also be set.</p>
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	<p>New in CS4. Setting this during PF_Cmd GLOBAL_SETUP puts the effect in the localized "Obsolete" folder in the Effects panel. Compare to PF_OutFlag_I_AM_OBSOLETE.</p>

TABLE 8: PF_OUTFLAGS2

Flag	Indicates
PF_OutFlag2_PPRO_DO_NOT_CLONE_SEQUENCE_DATA_FOR_RENDER	A flag that is currently only honored in Premiere Pro CS4.1 and later. We also anticipate it being used in the next version of Premiere Elements after version 7. This affects how Premiere Pro drives the plug-in using multithreading .
PF_OutFlag2_AUTOMATIC_WIDE_TIME_INPUT	<p>New in CS5. Set during PF Cmd GLOBAL SETUP. Requires setting of PF_OutFlag_WIDE_TIME_INPUT (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.</p> <p>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 using the new call PF_HaveInputsChangedOverTimeSpan before using the time-dependent data.</p>

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](#), using [PF_ADD_PARAM\(\)](#). Starting in 7.0, there is no more 128 parameter limit; 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	Data Type	Description
PF_Param_LAYER	PF_LayerDef	<p>Image and audio layers in the composition. All effects automatically have 1 layer parameter, param[0], the layer to which they are applied.</p> <p>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.</p> <p>NOTE: This is a reference to a layer which contains pixels and blips, not actual pixels and blips.</p>
PF_Param_SLIDER	long	No longer used.
PF_Param_FIX_SLIDER	PF_Fixed	<p>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.</p> <p>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.</p>
PF_Param_FLOAT_SLIDER	PF_FPLong	<p>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.</p> <p>PF_Param_FLOAT_SLIDERS also respond to slider flags discussed in Audio Filters.</p>
PF_Param_ANGLE	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_Boolean	PF_ParamFlag_CANNOT_INTERP is forced on for all checkboxes.
PF_Param_COLOR	PF_Pixel	24-bit RGB value (alpha is not used) that the user can choose either with the standard color picker or with an eye dropper tool.

TABLE 9: PARAMETER TYPES

Parameter Type	Data Type	Description
PF_Param_POINT	PF_Fixed	A two-dimensional point. 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. The new behavior assumes the point provides x and y values in destination layer space. Plug-ins which specify API versions before 12.1 will still get the old behavior.
PF_Param_POPUP	enum	List of choices. Specify a list of (read-only) pop-up entries ("Entry1 Entry2 Entry3"), and After Effects 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.
PF_Param_ARBITRARY	???	Custom data type. An arbitrary parameter 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.
PF_Param_PATH	PF_PathDef	Path parameters are references to masks applied to the same layer as the effect. Path parameter data cannot be accessed directly; use PF_PathQuerySuite and PF_PathDataSuite to manage and enquire about paths.
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)	

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](#)? 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](#)).

PARAM ZERO

The first parameter, `params[0]`, is the input image (a [PF_EffectWorld](#)) 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
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 or PF_Cmd_UPDATE_PARAMS_UI .
PF_ParamUIFlags	ui_flags	Specify a parameter's UI behavior before adding; only <code>PF_PUI_DISABLED</code> may be set during event handling.
short	ui_width	Width and height of the parameter's user interface (for non-standard parameters only).
short	ui_height	
PF_ParamType	param_type	Type of parameter.
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	flags	Specify a parameter's UI behavior before adding; only <code>PF_ParamFlag_COLLAPSE_TWIRLY</code> may be set during event handling.
PF_ParamDefUnion	u	A union of all possible parameter types . Only the type specified by <code>param_type</code> 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	Indicates
PF_PUI_TOPIC	Effect wants events for the parameter's title—the portion visible when any spinners are “twirled up”.
PF_PUI_CONTROL	You handle PF_Cmd_EVENTS for the control area (area that becomes invisible when you twirl up a parameter's spinner) in the ECP. You must also set PF_OutFlag_CUSTOM_UI during PF_Cmd_GLOBAL_SETUP. See Events for more details.
PF_PUI_STD_CONTROL_ONLY	No data stream will be associated with this parameter, and no keyframes will be available in the Timeline panel. This flag cannot be used with path, layer, or arbitrary data type params. You must also set PF_ParamFlag_SUPERVISE or the setting will never be used. This flag does not require that the PF_OutFlag_CUSTOM_UI flag be set, and can be used with arbitrary data. You can use After Effects' provided parameter interfaces for your arbitrary data. Set this flag and modify your parameter value when handling PF_Cmd_USER_CHANGED_PARAM .
PF_PUI_NO_ECW_UI	No ECP UI will be displayed.
PF_PUI_ECW_SEPARATOR	Put a thick line above this parameter.
PF_PUI_DISABLED	Disables (grays out) the parameter, usually in response to PF_Cmd_USER_CHANGED_PARAM .
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.

In addition to these flags, an effect parameter may be hidden or shown by using [AEGP_GetDynamicStreamFlags](#).

PARAMETER FLAGS

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

TABLE 12: PARAMETER FLAGS

Flag	Meaning
PF_ParamFlag_RESERVED1	Unused.
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_RESERVED2	Unused. Formerly PF_ParamFlag_WANTS_UPDATE.
PF_ParamFlag_RESERVED3	Unused. Formerly PF_ParamFlag_SEPERATE. Yes, we know that's not how you spell separate.
PF_ParamFlag_COLLAPSE_TWIRLY	Set this flag during PF Cmd USER CHANGED PARAM . This bit can now be set & cleared when handling PF Cmd UPDATE PARAMS UI and PF Cmd USER CHANGED PARAM 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 messages for this parameter. See Parameter Supervision 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 . This flag will not be honored unless PF_OutFlag2_PARAM_GROUP_START_COLLAPSED is set. Unfortunately, this was not honored in CS4, but it has been fixed in CS5.
PF_ParamFlag_USE_VALUE_FOR_OLD_PROJECTS	New in 7.0. 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.

TABLE 12: PARAMETER FLAGS

Flag	Meaning
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.
PF_ParamFlag_EXCLUDE_FROM_HAVE_INPUTS_CHANGED	New in CS5. Only relevant if the effect sets PF_OutFlag2_AUTOMATIC_WIDE_TIME_INPUT and will call PF_HaveInputsChangedOverTimeSpan

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
<code>world_flags</code>	Currently, the only flag is <code>PF_EffectWorldFlag_WRITEABLE</code> , which indicates that you are allowed to alter the image data of the world. Normally effects cannot alter input image data; only output.
<code>data</code>	Pointer to image data, stored as a <code>PF_PixelPtr</code> . Do not access directly; use the accessor macros . Image data in After Effects is always organized in 32-bpc chunky format — that is, sequential long words each contain Alpha, Red, Green, Blue from the low byte to the high byte.

TABLE 13: PF_EFFECTWORLD STRUCTURE

Item	Description
rowbytes	<p>The length, in bytes, of each row in the image's block of pixels. The block of pixels contains <code>height</code> lines each with <code>width</code> pixels followed by some bytes of padding. The <code>width</code> pixels (times four, because each pixel is four bytes long) plus optional extra padding adds up to <code>rowbytes</code> 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 <code>height</code> and <code>rowbytes</code>.</p> <p>NOTE: This value does not vary based on whether field rendering is active.</p> <p>NOTE: Input and output worlds with the same dimensions can use different <code>rowbytes</code> values.</p>
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.
pix_aspect_ratio	The pixel aspect ratio expressed as a <code>PF_Rational</code> . NOTE: Effects can use this value for checked out layers, but must use <code>PF_InData.pixel_aspect_ratio</code> for the layer to which they're applied. Sorry.
platform_ref	<p>No longer used in CS5.</p> <p>Platform-specific reference information. On Windows, this contains an opaque value. Under Mac OS, <code>PF_GET_PLATFORM_REFS</code> provides a <code>CGrafPtr</code> and a <code>GDeviceHandle</code> from a <code>PF_EffectWorld</code>.</p> <p>NOTE: You cannot acquire a <code>platform_ref</code> during PF_Cmd GLOBAL SETUP, as there isn't any output context yet. Patience, my pet.</p>

WHAT HAPPENED TO PF_WORLDS?

Grizzled API veterans may remember that layers used to be called `PF_Worlds`, and that these structures bore a strong similarity to (in that they were simply wrappers for) Mac OS `GWorlds`. We renamed `PF_Worlds` to `PF_EffectWorlds` in 6.0, to emphasize that you can no longer “cheat” and use them as `GWorlds`, and that pixels are represented by different structures in different parts of the API, so you should be relying on [accessor](#) and manipulator functions instead of tinkering directly with the `PF_World`...err...we mean `PF_EffectWorld`.

ROWBYTES IN PF_EFFECTWORLDS

Don't assume that you can get to the next scanline of a [PF_EffectWorld](#) using `(width * sizeof(current_pixel_type)) + 4`, or whatever; use the [PF_EffectWorld](#)'s [rowbytes](#) 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](#)'s [rowbytes](#), 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](#) 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

As of 5.0, After Effects supported 16 bit-per-channel color; as of 7.0, After Effects supports 32 bit-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	<p>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.</p> <pre>PF_GET_PIXEL_DATA16 (PF_EffectWorld *wP, PF_PixelPtr pP0, PF_Pixel16 **outPP);</pre>
PF_GET_PIXEL_DATA8	<p>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.</p> <pre>PF_GET_PIXEL_DATA8 (PF_EffectWorld *wP, PF_PixelPtr pP0, PF_Pixel8 **outPP);</pre>

Think of [PF_GET_PIXEL_DATA16](#) and [PF_GET_PIXEL_DATA8](#) 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
<code>PF_Err_NONE</code>	Success.
<code>PF_Err_OUT_OF_MEMORY</code>	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.
<code>PF_Err_INTERNAL_STRUCT_DAMAGED</code>	Problems using a data structure.
<code>PF_Err_INVALID_INDEX</code>	Problems finding/using array member.
<code>PF_Err_UNRECOGNIZED_PARAM_TYPE</code>	Problem with parameter data.
<code>PF_Err_INVALID_CALLBACK</code>	Problems accessing function through pointer.
<code>PF_Err_BAD_CALLBACK_PARAM</code>	Problems using a parameter passed to a callback.
<code>PF_Interrupt_CANCEL</code>	User cancelled rendering.
<code>PF_Err_CANNOT_PARSE_KEYFRAME_TEXT</code>	Return this from <code>PF_Arbitrary_SCAN_FUNC</code> 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 `nonui` 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`. 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 reusable 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 be 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](#).

VERSIONING

`WhizBangSuite1` may provide a `FooBar()` function which takes two arguments, and `WhizBangSuite2>FooBar()` may take three. Though each new version of a suite supersedes 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

Always use After Effects memory allocation functions. In low-memory conditions (such as during RAM preview), it’s very important that plug-ins not compete with After Effects for OS memory, and deal gracefully with out-of-memory conditions. Failing to use our functions can cause lock-ups, crashes, and tech support calls. Don’t do that.

If you’re wrapping existing C++ code, overloading `new` and `delete` to use our functions will save substantial reimplementation. On Windows, derive all classes from a common base class which implements `new` and `delete`.

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.
host_lock_handle	Locks a handle. Replaces PF_LOCK_HANDLE.
host_unlock_handle	Unlocks a handle. Replaces PF_UNLOCK_HANDLE.
host_dispose_handle	Frees a handle. Replaces PF_DISPOSE_HANDLE.
host_get_handle_size	Returns the size, in bytes, of the reallocatable block whose handle is passed in. Replaces PF_GET_HANDLE_SIZE.
host_resize_handle	Resizes a handle. Replaces PF_RESIZE_HANDLE.

IMAGE BUFFER MANAGEMENT FUNCTIONS

Use these functions to create and destroy [PF_EffectWorlds](#).

TABLE 17: PF_WORLDSUITE1

Function	Description
new_world	<p>Creates a new PF_EffectWorld.</p> <pre>PF_Err new_world(PF_ProgPtr effect_ref, long width, long height, PF_NewWorldFlags flags, PF_EffectWorld *world);</pre> <p>PF_NewWorldFlags can be:</p> <p>PF_NewWorldFlag_NONE PF_NewWorldFlag_CLEAR_PIXELS PF_NewWorldFlag_DEEP_PIXELS</p>
dispose_world	<p>Disposes of a PF_EffectWorld.</p> <pre>PF_Err dispose_world(PF_ProgPtr effect_ref, PF_EffectWorld *world);</pre>

SAME MEANING, DIFFERENT FLAGS

When creating worlds, pass `PF_NewWorldFlags` to set the world properties. When investigating *existing* worlds, check them for [PF_EffectWorldFlags](#). They are different flag sets. No, we're not sure why either.

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: ITERATION SUITE FUNCTIONS

Function	Purpose
<code>iterate</code>	<p>Iterates across pixels from a source image, alters them, and populates a destination image.</p> <p>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.</p> <p>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 <code>dst</code>) 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.</p> <pre>iterate(PF_InData *in_data, long progress_base, long progress_final, PF_EffectWorld *src, const PF_Rect *area, void *refcon, PF_Err (*pix_fn)(void *refcon, long x, long y, PF_Pixel *in, PF_Pixel *out), PF_EffectWorld *dst);</pre>

TABLE 18: ITERATION SUITE FUNCTIONS

Function	Purpose
iterate_origin	<p>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.</p> <pre> iterate_origin(PF_InData *in_data, long progress_base, long progress_final, PF_EffectWorld *src, const PF_Rect *area, const PF_Point *origin, void *refcon, PF_Err (*pix_fn)(void *refcon, long x, long y, PF_Pixel *in, PF_Pixel *out), PF_EffectWorld *dst); </pre>
iterate_lut	<p>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.</p> <pre> iterate_lut(PF_InData *in_data, long prog_base, long prog_final, PF_EffectWorld *src, const PF_Rect *area, unsigned char *a_lut0, unsigned char *r_lut0, unsigned char *g_lut0, unsigned char *b_lut0, PF_EffectWorld *dstP); </pre>

TABLE 18: ITERATION SUITE FUNCTIONS

Function	Purpose
<code>iterate_origin_non_clip_src</code>	<p>Allows for iteration across pixels outside the intersection of the source and destination layers. For these pixels, you will be passed a <code>PF_Pixel</code> with values {0,0,0,0}.</p> <pre> iterate_origin_non_clip_src(PF_InData *in_data, long progress_base, long progress_final, PF_EffectWorld *src, const PF_Rect *area, const PF_Point *origin, void *refcon, PF_Err (*pix_fn)(void *refcon, long x, long y, PF_Pixel *in, PF_Pixel *out), PF_EffectWorld *dst); </pre>
<code>iterate_generic</code>	<p>If you want to do something once per available CPU, this is the function to use (pass <code>PF_Iterations_ONCE_PER_PROCESSOR</code> for iterationsL). Only call abort and progress functions from thread index 0.</p> <p>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.</p> <pre> iterate_generic(long iterationsL, void *refconPV, PF_Err (*fn_func)(void *refconPV, long long i, long itrL); thread_idxL, </pre>

GRAPHICS UTILITY SUITES

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

TRANSFORM WORLDS

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	<p>Composite a rectangle from one PF_EffectWorld into another, using one of After Effects' transfer modes.</p> <pre>PF_Err composite_rect (PF_ProgPtr effect_ref, PF_Rect *src_rect, long src_opacity, PF_EffectWorld *src_world, long dst_x, long dst_y, PF_Field field_rdr, PF_XferMode xfer_mode, PF_EffectWorld *dst);</pre> <p>field_rdr can be upper, lower or both.</p> <p>xfer_mode is one of the following:</p> <p>PF_Xfer_COPY PF_Xfer_BEHIND PF_Xfer_IN_FRONT</p>
blend	<p>Blends two images, alpha-weighted. Does not deal with different-sized sources, though the destination may be either PF_EffectWorld.</p> <pre>PF_Err blend (PF_ProgPtr effect_ref, const PF_EffectWorld *src1, const PF_EffectWorld *src2, PF_Fixed ratio, PF_EffectWorld *dst);</pre>

TABLE 19: PF_WORLDTRANSFORMSUITE1

Function	Purpose
convolve	<p>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), or pass 0 to convolve the entire image. Do not use if the source is the destination. Describe the convolution using kernel flags.</p> <pre>PF_Err convolve(PF_EffectWorld *src, const PF_Rect *area, PF_KernelFlags flags, long kernel_size, void *a_kernel, void *r_kernel, void *g_kernel, void *b_kernel, PF_EffectWorld *dst);</pre>
copy	<p>Copies a region from one PF_EffectWorld to another, preserving alpha (unlike the Mac OS CopyBits).</p> <pre>PF_Err copy (PF_EffectWorld *src, PF_EffectWorld *dst, PF_Rect *src_r, PF_Rect *dst_r);</pre>
copy_hq	A higher fidelity version of the above (using the same parameters).

TABLE 19: PF_WORLDTRANSFORMSUITE1

Function	Purpose
transfer_rect	<p>Blends using a transfer mode, with an optional mask.</p> <pre>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, long dest_x, long dest_y, PF_EffectWorld *dst_world);</pre>
transform_world	<p>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.</p> <p>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.</p> <pre>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, long num_matrices, Boolean src2dst_matrix, const PF_Rect *dest_rect, PF_EffectWorld *dst_world);</pre>

KERNEL FLAGS

Many functions 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. <pre>PF_Err fill (PF_ProgPtr effect_ref, const PF_Pixel *color, const PF_Rect *dst_rect, PF_EffectWorld *world);</pre>
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. <pre>PF_Err premultiply (long forward, PF_EffectWorld *dst);</pre>
premultiply_color	Converts to (and from) having r, g, and b color values premultiplied with any color to represent the alpha channel. <pre>PF_Err premultiply_color (PF_ProgPtr effect_ref, PF_EffectWorld *src, PF_Pixel *color, long forward, PF_EffectWorld *dst);</pre>
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. <pre>PF_Err nn_sample (PF_ProgPtr effect_ref, PF_Fixed x, PF_Fixed y, const PF_SampPB *params, PF_Pixel *dst_pixel);</pre>
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. <pre>PF_Err subpixel_sample (PF_ProgPtr effect_ref, PF_Fixed x, PF_Fixed y, const PF_SampPB *params, PF_Pixel *dst_pixel);</pre>
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.

TABLE 22: PF_SAMPLINGSUITE FUNCTIONS (MULTIPLE SUITES)

Function	Purpose
area_sample	<p>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.</p> <pre>PF_Err area_sample (PF_ProgPtr effect_ref, PF_Fixed x, PF_Fixed y, const PF_SampPB *params, PF_Pixel *dst_pixel);</pre> <p>NOTE: Areas outside the boundaries of the layer are considered the same as zero alpha, for sampling purposes.</p>
area_sample16	Same as above, but takes a PF_Pixel16* dst_pixel.

TABLE 23: PF_BATCHSAMPLINGSUITE1

Function	Purpose
begin_sampling	<p>Your effect is going to perform some batch sampling; After Effects will perform setup tasks to optimize your sampling.</p> <pre>PF_Err (*begin_sampling)(PF_ProgPtr effect_ref, PF_Quality qual, PF_ModeFlags mf, PF_SampPB *params);</pre>
end_sampling	<p>Tells After Effects you're done sampling.</p> <pre>PF_Err (*end_sampling)(PF_ProgPtr effect_ref, PF_Quality qual, PF_ModeFlags mf, PF_SampPB *params);</pre>

TABLE 23: PF_BATCHSAMPLINGSUITE1

Function	Purpose
get_batch_func	Obtains a pointer to After Effects' batch sampling function (highly optimized). <pre>PF_Err (*get_batch_func)(PF_ProgPtr effect_ref, PF_Quality quality, PF_ModeFlags mode_flags, const PF_SampPB *params, PF_BatchSampleFunc *batch);</pre>
get_batch_func16	Obtains a pointer to After Effects' 16-bpc batch sampling function (also highly optimized). <pre>PF_Err (*get_batch_func16)(PF_ProgPtr effect_ref, PF_Quality quality, PF_ModeFlags mode_flags, const PF_SampPB *params, PF_BatchSample16Func *batch);</pre>

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.

TABLE 24: PF_ANSICALLBACKSSUITE1

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 $\sqrt{x^2 + y^2}$. 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.
<i>(while not strictly math functions, these emulate ANSI functionality)</i>	
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](#), 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	<p>Enumerate your plug-in's parameters to After Effects during PF_Cmd PARAM SETUP, using multiple calls to this function.</p> <p>Note: Failing to completely clear out a <code>PF_ParamDef</code> prior to <code>PF_ADD_PARAM()</code> can cause many problems. Always use <code>AEFX_CLR_STRUCT</code> before adding parameters.</p> <pre>PF_Err PF_ADD_PARAM (PF_InData *in_data, PF_ParamIndex index, PF_ParamDefPtr def);</pre>
PF_ABORT	<p>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.</p> <pre>PF_Err PF_ABORT (PF_InData *in_data);</pre>
PF_PROGRESS	<p>Displays a progress bar during processing; <code>current</code> and <code>total</code> 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 <code>total</code> is 0, <code>PF_ABORT</code> is used instead (presenting the user with different choices).</p> <pre>PF_Err PF_PROGRESS (PF_InData *in_data, long current, long total);</pre>

TABLE 25: INTERACTION CALLBACKS

Function	Purpose
PF_CHECKOUT_PARAM	<p>Obtains parameter values, or the source video layer, at a specified time. After Effects makes caching decisions based on the checkout state of parameters.</p> <p>Allocate a new PF_ParamDef 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.</p> <p>Do not check out layer parameters during UI event handling.</p> <pre>PF_Err PF_CHECKOUT_PARAM (PF_InData *in_data, PF_ParamIndex index, long what_time, long step, long time_scale, PF_ParamDef *param);</pre> <p>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.</p>
PF_CHECKIN_PARAM	<p>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 will no longer be valid.</p> <pre>PF_Err PF_CHECKIN_PARAM (PF_InData *in_data, PF_ParamDef *param);</pre>
PF_REGISTER_UI	<p>Register a custom user interface element. See Events.</p> <pre>PF_Err PF_REGISTER_UI (PF_InData *in_data, PF_CustomUIInfo *cust_info);</pre>

TABLE 25: INTERACTION CALLBACKS

Function	Purpose
PF_CHECKOUT_LAYER_AUDIO	<p>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.</p> <pre> PF_Err PF_CHECKOUT_LAYER_AUDIO (PF_InData *in_data, PF_ParamIndex index, long start_time, long duration, unsigned long time_scale, PF_UFixed rate, long bytes_per_sample, long num_channels, long fmt_signed, PF_LayerAudio *audio); </pre>
PF_CHECKIN_LAYER_AUDIO	<p>Balance all calls to PF_CHECKOUT_LAYER_AUDIO, regardless of error conditions, with matching calls to PF_CHECKIN_LAYER_AUDIO.</p> <pre> PF_Err PF_CHECKIN_LAYER_AUDIO (PF_InData *in_data, PF_LayerAudio audio); </pre>
PF_GET_AUDIO_DATA	<p>Returns information about the PF_LayerAudio.</p> <p>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 <i>alter</i> audio, write an audio filter.</p> <pre> PF_Err PF_GET_AUDIO_DATA (PF_InData *in_data, PF_LayerAudio audio, PF_SndSamplePtr *data0, long *num_samples0, PF_UFixed *rate0, long *bytes_per_sample0, long *num_channels0, long *fmt_signed0); </pre>

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\[0\]](#)) of effect[n]. On the other hand, when a normal effect checks out a layer using [PF_CHECKOUT_LAYER_AUDIO](#), it receives the raw (un-effected) source layer, regardless of its order. However, when a [SmartFX](#) 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 its output every frame despite the static input layer, you'd need to set

[PF_Outflag_NON_PARAM_VARY](#).

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](#). 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](#) 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;  
  
x *= in_data>downsample_x.den /  
(float)in_data>downsample_x.num;  
  
y *= in_data>downsample_y.den /  
(float)in_data>downsample_y.num;
```

2) 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.

3) 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.

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](#) before adding it during `PF_Cmd_PARAMS_SETUP`. Whenever it is changed, you will receive [PF_Cmd_USER_CHANGED_PARAM](#). 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

During `PF_Cmd_UPDATE_PARAMS_UI`, you may only change the appearance and enable state of parameters. Use [PF_UpdateParamUI\(\)](#) from [PF_ParamUtilsSuite1](#) 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.

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`.

BEAUTY IS ONLY UI DEEP

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`. Note that `PF_ChangeFlag_CHANGED_VALUE` isn't supported for layer parameters. This change will be undoable by the user. Note: prior to the 5.5 release, effects had to set [PF_OutFlag_REFRESH_UI](#) in `PF_OutData>outflags` to force the UI to reflect your changes. While harmless, this is no longer necessary.

WHEN YOU CAN CHANGE PARAMETER VALUES

A parameter's value (not just UI) can be modified during [PF_Cmd_EVENT](#) (any event) and during [PF_Cmd_USER_CHANGED_PARAM](#). After Effects will not honor changes made at other times.

PARAMETER UTILITY SUITE

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 26: PF_PARAMUTILSSUITE1

Function	Purpose
PF_UpdateParamUI	<pre>PF_UpdateParamUI(PF_ProgPtr effect_ref, PF_ParamIndex param_index, const PF_ParamDef *defP);</pre> <p>Force After Effects to refresh the parameter's UI, in the effect controls palette.</p> <p>NOTE: Never pass param[0] to this function.</p>
PF_GetCurrentState	<pre>PF_GetCurrentState(PF_ProgPtr effect_ref, PF_State *stateP);</pre> <p>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).</p>
PF_HasParamChanged	<pre>PF_HasParamChanged(PF_ProgPtr effect_ref, const PF_State *stateP, PF_ParamIndex param_index, PF_Boolean *changedPB);</pre> <p>Given a PF_State, passes back true if any of the tested parameters differ from the saved state. Pass PF_ParamIndex_CHECK_ALL to check every parameter, PF_ParamIndex_CHECK_ALL_EXCEPT_LAYER_PARAMS to omit layers.</p>

TABLE 26: PF_PARAMUTILSSUITE1

Function	Purpose
PF_HaveInputsChangedOverTimeSpan	<pre>PF_HaveInputsChangedOverTimeSpan(PF_ProgPtr effect_ref, const PF_State *stateP, const A_Time *startPT0, const A_Time *durationPT0, PF_Boolean *changedPB);</pre> <p>Very useful for <i>validating sequence data</i>. Given a PF_State, passes back true if any of the tested parameters differ from the saved state over the time period specified. Parameters can be omitted by setting PF_ParamFlag_EXCLUDE_FROM_HAVE_INPUTS_CHANGED.</p> <p>Requires PF_OutFlag2_AUTOMATIC_WIDE_TIME_INPUT to be set by the effect. If validating a cache for use during a render, this call must happen during one of the rendering PF_Cmds (PF_Cmd_FRAME_SETUP, PF_Cmd_RENDER, PF_Cmd_FRAME_SETDOWN, PF_Cmd_SMART_PRE_RENDER, PF_Cmd_SMART_RENDER).</p> <p>This call also implicitly tells the host that the effect's output for the current frame depends on the specified frames, so if something changes upstream, the host's caches will be properly invalidated automatically.</p>
PF_IsIdenticalCheckout	<pre>PF_IsIdenticalCheckout(PF_ProgPtr effect_ref, PF_ParamIndex param_index, long what_time1, long time_step1, A_u_long time_scale1, long what_time2, long time_step2, A_u_long time_scale2, PF_Boolean *identicalPB);</pre> <p>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.</p>

TABLE 26: PF_PARAMUTILSSUITE1

Function	Purpose
PF_FindKeyframeTime	<pre>PF_FindKeyframeTime(PF_ProgPtr effect_ref, PF_ParamIndex param_index, long what_time, A_u_long time_scale, PF_TimeDir time_dir, PF_Boolean *foundPB, PF_KeyIndex *key_indexP0, long *key_timeP0, A_u_long *key_timescaleP0);</pre> <p>Searches (in the specified direction) for the next keyframe in the parameter's stream. The last three parameters are optional.</p>
PF_GetKeyframeCount	<pre>PF_GetKeyframeCount(PF_ProgPtr effect_ref, PF_ParamIndex param_index, PF_KeyIndex *key_countP);</pre> <p>Returns the number of keyframes in the parameter's stream.</p>
PF_CheckoutKeyframe	<pre>PF_CheckoutKeyframe(PF_ProgPtr effect_ref, PF_ParamIndex param_index, PF_KeyIndex key_index, long *key_timeP0, A_u_long *key_timescaleP0, PF_ParamDef *paramP0);</pre> <p>Checks a keyframe for the specified parameter out of our keyframe database. <code>param_index</code> is zero-based. You can request time, timescale, or neither; useful if you're performing your own motion blur.</p>
PF_CheckinKeyframe	<pre>PF_CheckinKeyframe(PF_ProgPtr effect_ref, PF_ParamDef *paramP);</pre> <p>All calls to <code>PF_CheckoutKeyframe</code> must be balanced with this check-in, or pain will ensue.</p>
PF_KeyIndexToTime	<pre>PF_KeyIndexToTime(PF_ProgPtr effect_ref, PF_ParamIndex param_index, PF_KeyIndex key_indexP, long *key_timeP, A_u_long *key_timescaleP);</pre> <p>Returns the time (and timescale) of the specified keyframe.</p>

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 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_HaveInputsChangedOverTimeSpan`](#)(start=0, duration=N-1) 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`](#) set), including layer parameters (including [`param\[0\]`](#)) 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. Set [PF_OutFlag_SEQUENCE_DATA_NEEDS_FLATTENING](#) during [PF_Cmd_GLO-BAL_SETUP](#), and in your PiPL.

Upon receiving [PF_Cmd_SEQUENCE_FLATTEN](#), 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](#) 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 {
    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](#), 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 27: ARBITRARY DATA SELECTORS

Selector	Response
<i>PF_Arbitrary_NEW_FUNC</i>	Allocate, populate, and return a handle to a new instance of your arb data.
<i>PF_Arbitrary_DISPOSE_FUNC</i>	Free and destroy an instance of your arbitrary data type.
<i>PF_Arbitrary_COPY_FUNC</i>	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.
<i>PF_Arbitrary_FLAT_SIZE_FUNC</i>	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.
<i>PF_Arbitrary_FLATTEN_FUNC</i>	Flatten the instance you’re passed, and place it in the supplied buffer. The buffer will be the size you reported in response to <i>PF_Arbitrary_FLAT_SIZE_FUNC</i> .
<i>PF_Arbitrary_UNFLATTEN_FUNC</i>	Unpack the buffer into an instance of your arbitrary data type, and put in the handle which you’ve been passed.

TABLE 27: ARBITRARY DATA SELECTORS

Selector	Response
<i>PF_Arbitrary_INTERP_FUNC</i>	<p>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 <code>float</code> indicating where, between 0 and 1, your interpreted value should be.</p> <p>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.</p> <p>NOTE: Never check out parameters if the in_data>effect_ref is NULL.</p>
<i>PF_Arbitrary_COMPARE_FUNC</i>	<p>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 <code>PF_ArbCompareResult</code> (see <code>AE_Effect.h</code>) to indicate whether the first was equal to, less than, more than, or simply not equal to the second.</p>
<i>PF_Arbitrary_PRINT_SIZE_FUNC</i>	<p>Indicate the buffer size you require for printing your parameter's current values by setting <code>print_sizePLu</code> (member of <code>print_size_func_params</code>, part of the <code>PF_ArbParamsExtra</code> structure).</p>
<i>PF_Arbitrary_PRINT_FUNC</i>	<p>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.</p>
<i>PF_Arbitrary_SCAN_FUNC</i>	<p>Given a buffer of text data (often from the system clipboard), parse it into your arbitrary data format.</p>

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](#). 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](#). 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 28: PF_EFFECTUISUITE

Function	Purpose
PF_SetOptionsButtonName	<p>Changes the text on the options button in the effect controls palette. NOTE: This must be called during PF_Cmd_PARAM_SETUP.</p> <pre>PF_SetOptionsButtonName(PF_ProgPtr effect_ref, const A_char *nameZ);</pre> <p>nameZ may be up to A_char[31] in length.</p>

PF_APPSUIE

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](#) for custom UI in effects, use these calls to integrate seamlessly into the After Effects UI. We used to provide additional calls via PFAppearanceSuite, but then ADM went away in CS3 (8.0). Back to basics.

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 29: PF_APPSUIE4

Function	Purpose
PF_AppGetBgColor	Retrieves the current background color. <pre>PF_AppGetBgColor(PF_App_Color *bg_colorP);</pre>
PF_AppGetColor	Retrieves the specified color. See AE_EffectSuites.h for a complete enumeration of available PF_App_Color values; basically any color in After Effects' UI can be retrieved. <pre>PF_AppGetColor(PF_App_ColorType color_type, PF_App_Color *app_colorP);</pre>
PF_GetPersonalInfo	Retrieves the user's registration information. <pre>PF_GetPersonalInfo(PF_AppPersonalTextInfo *ptiP);</pre> <pre>typedef struct PF_AppPersonalTextInfo { char name[PF_APP_MAX_PERS_LEN + 1]; char org[PF_APP_MAX_PERS_LEN + 1]; char serial_str[PF_APP_MAX_PERS_LEN+1]; } PF_AppPersonalTextInfo;</pre>
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. <pre>PF_GetFontStyleSheet(PF_FontStyleSheet sheet, PF_FontName *font_nameP0, short *font_numPS0, short *sizePS0, short *stylePS0);</pre>
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). <pre>PF_SetCursor(PF_CursorType cursor);</pre>

TABLE 29: PF_APPSUITE4

Function	Purpose
PF_IsRenderEngine	<p>Returns TRUE if After Effects is running in watched folder mode, or is a render engine installation.</p> <pre>PF_IsRenderEngine(PF_Boolean *render_enginePB);</pre> <p>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.</p> <p>In 5.0, this function returns true in render engine installs, and when After Effects is watching a folder.</p>
PF_AppColorPickerDialog	<p>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.</p> <pre>PF_AppColorPickerDialog(const A_char *dialog_titleZ0, const PF_PixelFloat *sample_colorP, PF_PixelFloat *result_colorP);</pre>
PF_GetMouse	<p>New in 7.0. Returns the position of the mouse in the custom UI coordinate space.</p> <pre>PF_GetMouse(PF_Point *pointP);</pre>
PF_InvalidateRect	<p>New in CS5. Queue up a redraw 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.</p> <pre>PF_InvalidateRect(const PF_ContextH contextH, const PF_Rect* rectP0);</pre>
PF_ConvertLocalToGlobal	<p>New in 7.0. Converts from the custom UI coordinate system to global screen coordinates. Use only during custom UI event handling.</p> <pre>PF_ConvertLocalToGlobal(const PF_Point *localP, PF_Point *globalP);</pre>

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 30: AE_ADVAPPSUITE2

Function	Purpose
PF_SetProjectDirty	Tells After Effects that the project has been changed since it was last saved. <code>PF_SetProjectDirty(void);</code>
PF_SaveProject	Saves the project to the current path. To save the project elsewhere, use AEGP_SaveProjectToPath() . <code>PF_SaveProject(void);</code>
PF_SaveBackgroundState	Stores the background state (After Effects' position in the stacking order of open applications and windows). <code>PF_SaveBackgroundState(void);</code>
PF_ForceForeground	Brings After Effects to the front of all currently open applications and windows. <code>PF_ForceForeground(void);</code>
PF_RestoreBackgroundState	Puts After Effects back where it was, in relation to other applications and windows. <code>PF_RestoreBackgroundState(void);</code>
PF_RefreshAllWindows	Forces all After Effects windows to update. <code>PF_RefreshAllWindows(void);</code>
PF_InfoDrawText	Writes text into the After Effects info palette. <code>PF_InfoDrawText(const char *line1Z0, const char *line2Z0);</code>
PF_InfoDrawColor	Draws the specified color in the After Effects info palette (alpha is ignored). <code>PF_InfoDrawColor(PF_Pixel color);</code>
PF_InfoDrawText3	Writes three lines of text into the After Effects info palette. <code>PF_InfoDrawText3(const char *line1Z0, const char *line2Z0, const char *line3Z0);</code>

TABLE 30: AE_ADVAPPSUITE2

Function	Purpose
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. <pre>PF_InfoDrawText3Plus(const char *line1Z0, const char *line2_jrZ0, const char *line2_jlZ0, const char *line3_jrZ0, const char *line3_jlZ0);</pre>
PF_AppendInfoText	Appends characters to the currently-displayed info text. <pre>PF_AppendInfoText(const char *appendZ0);</pre>

FORMATTING TIME

PF_AdvTimeSuite provides several functions to match how After Effects displays time. In fact, these are the same functions we use internally.

TABLE 31: PF_ADVTIMESUITE1

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. <pre>PF_FormatTimeActiveItem(long time_valueUL, unsigned long time_scaleL, PF_Boolean durationB, char *time_buf);</pre>
PF_FormatTime	Contextualizes the formatted time string for the given PF_InData and PF_EffectWorld (i.e., layer time). <pre>PF_FormatTime(PF_InData *in_data, PF_EffectWorld *world, long time_valueUL, unsigned long time_scaleL, PF_Boolean durationB, char *time_buf);</pre>

TABLE 31: PF_ADVTIMEsuite1

Function	Purpose
PF_FormatTimePlus	Allows you to select composition or layer time. <pre>PF_FormatTimePlus(PF_InData *in_data, PF_EffectWorld *world, long time_valueUL, unsigned long time_scaleL, PF_Boolean comp_timeB, PF_Boolean durationB, char *time_buf);</pre>
PF_GetTimeDisplayPref	Returns the starting frame number (specified by the user in composition settings). <pre>PF_GetTimeDisplayPref(PF_TimeDisplayPref *tdp, long *starting_num);</pre>

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](#).

TABLE 32: PF_ADVITEMsuite1

Function	Purpose
PF_MoveTimeStep	Moves current time num_stepsL in the specified direction. <pre>PF_MoveTimeStep(PF_InData *in_data, PF_EffectWorld *world, PF_Step time_dir, long num_stepsL);</pre>
PF_MoveTimeStep ActiveItem	Moves num_stepsL in the specified direction, for the active item. <pre>PF_MoveTimeStepActiveItem(PF_Step time_dir, long num_stepsL);</pre>
PF_TouchActiveItem	Tells After Effects that the active item must be updated. <pre>PF_TouchActiveItem (void);</pre>

TABLE 32: PF_ADVITEMSUITE1

Function	Purpose
PF_ForceRerender	Forces After Effects to rerender the current frame. <pre>PF_ForceRerender(PF_InData *in_data, PF_EffectWorld *world);</pre>
PF_EffectIsActiveOrEnabled	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). <pre>PF_EffectIsActiveOrEnabled(PF_ContextH contextH, PF_Boolean *enabledPB);</pre>

ACCESSING AUXILIARY CHANNEL DATA

Some file types contain more than just pixel data; use [PF_ChannelSuite](#) to determine whether such information is present, and the macros in `AE_ChannelSuites.h` to retrieve it in the format you need.

TABLE 33: PF_CHANNELSUITE1

Function	Purpose
PF_GetLayerChannelCount	Retrieves the number of auxiliary channels associated with the indexed layer. <pre>PF_GetLayerChannelCount(PF_ProgPtr effect_ref, PF_ParamIndex param_index, long *num_channelsPL);</pre>
PF_GetLayerChannelIndexedRefAndDesc	Retrieves (by index) a reference to, and description of, the specified channel. <pre>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);</pre>

TABLE 33: PF_CHANNELSUITE1

Function	Purpose																				
PF_GetLayerChannel TypedRefAndDesc	<p>Retrieves an auxiliary channel by type. Returned information is valid only if foundPB returns TRUE.</p> <pre>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);</pre> <p>PF_DataType will be one of the following:</p> <table> <tr><td>PF_DataType_FLOAT</td><td>34 bytes</td></tr> <tr><td>PF_DataType_DOUBLE</td><td>38 bytes</td></tr> <tr><td>PF_DataType_LONG</td><td>34 bytes</td></tr> <tr><td>PF_DataType_SHORT</td><td>32 bytes</td></tr> <tr><td>PF_DataType_FIXED_16_16</td><td>34 bytes</td></tr> <tr><td>PF_DataType_CHAR</td><td>31 byte</td></tr> <tr><td>PF_DataType_U_BYTE</td><td>31 byte</td></tr> <tr><td>PF_DataType_U_SHORT</td><td>32 bytes</td></tr> <tr><td>PF_DataType_U_FIXED_16_16</td><td>34 bytes</td></tr> <tr><td>PF_DataType_RGB</td><td>3 bytes</td></tr> </table> <p>PF_ChannelType will be one of the following:</p> <pre>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</pre>	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_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_CheckoutLayerChannel	<p>Retrieves the PF_ChannelChunk containing the data associated with the given PF_ChannelRefPtr.</p> <pre>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);</pre>																				

TABLE 33: PF_CHANNELSUITE1

Function	Purpose
PF_CheckinLayerChannel	Checks in the PF_ChannelChunk. Always, always, always check the data back in. <pre>PF_CheckinLayerChannel(PF_ProgPtr effect_ref, PF_ChannelRefPtr channel_refP, PF_ChannelChunk *channel_chunkP);</pre>

COLOR PARAMETERS AND FLOATING POINT COLOR

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 34: PF_COLORPARAMSUITE1

Function	Purpose
PF_GetFloatingPointColorFromColorDef	PF_GetFloatingPointColorFromColorDef(<pre>PF_ProgPtr effect_ref, const PF_ParamDef *color_defP, PF_PixelFloat *fp_colorP);</pre>

MOTION BLUR

Effects handle their own motion blur, using PF_InData>[shutter_angle](#) along with PF_InData>[shutter_phase](#). The plug-in must set [PF_OutFlag_I_USE_SHUTTER_ANGLE](#) so After Effects knows it needs this information. They must [check out](#) 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](#). 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](#) and [PF_PathDataSuite](#). 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](#) to manipulate paths. See "[cheating](#)". 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 35: 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 36: 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, long *num_segmentsPL);
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, long which_pointL, PF_PathVertex *vertexP);
PF_PathPrepareSegLength	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, long which_segL, long frequencyL, PF_PathSegPrepPtr *lengthPrepPP);
PF_PathGetSegLength	Retrieves the length of the given segment. PF_PathGetSegLength(PF_ProgPtr effect_ref0, PF_PathOutlinePtr pathP, long which_segL, PF_PathSegPrepPtr *lengthPrepP0, PF_FpLong *lengthPF);

TABLE 36: PF_PATHDATASUITE1

Function	Description
PF_PathEvalSegLength	<p>Retrieves the location of a point lengthF along the given path segment.</p> <pre>PF_PathEvalSegLength(PF_ProgPtr effect_ref0, PF_PathOutlinePtr pathP, PF_PathSegPrepPtr *lengthPrepPP0, long which_segL, PF_FpLong lengthF, PF_FpLong *x, PF_FpLong *y);</pre>
PF_PathEvalSegLengthDeriv1	<p>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.</p> <pre>PF_PathEvalSegLengthDeriv1(PF_ProgPtr effect_ref0, PF_PathOutlinePtr pathP, PF_PathSegPrepPtr *lengthPrepPP0, long which_segL, PF_FpLong lengthF, PF_FpLong *x, PF_FpLong *y, PF_FpLong *deriv1x, PF_FpLong *deriv1y);</pre>
PF_PathCleanupSegLength	<p>Call this when you're finished evaluating that segment length, so After Effects can properly clean up the PF_PathSegPrepPtr.</p> <pre>PF_PathCleanupSegLength(PF_ProgPtr effect_ref0, PF_PathOutlinePtr pathP, long which_segL, PF_PathSegPrepPtr *lengthPrepPP);</pre>
PF_PathIsInverted	<p>Returns TRUE if the path is inverted.</p> <pre>PF_PathIsInverted(PF_ProgPtr effect_ref, PF_PathID unique_id, PF_Boolean *invertedB);</pre>

TABLE 36: PF_PATHDATASUITE1

Function	Description
PF_PathGetMaskMode	<p>Retrieves the mode for the given path.</p> <pre>PF_PathGetMaskMode(PF_ProgPtr effect_ref, PF_PathID unique_id, PF_MaskMode *modeP);</pre> <p>Mask mode is one of the following:</p> <pre>PF_MaskMode_NONE PF_MaskMode_ADD PF_MaskMode_SUBTRACT PF_MaskMode_INTERSECT PF_MaskMode_LIGHTEN PF_MaskMode_DARKEN PF_MaskMode_DIFFERENCE PF_MaskMode_ACCUM</pre>
PF_PathGetName	<p>Retrieves the name of the path (up to PF_MAX_PATH_NAME_LEN long).</p> <pre>PF_PathGetName(PF_ProgPtr effect_ref, PF_PathID unique_id, char *nameZ);</pre>

PF_PATHQUERYSUITE

This suite is used to identify and access the paths associated with the effect's source layer.

TABLE 37: PF_PATHQUERYSUITE

Function	Purpose
PF_NumPaths	<p>Retrieves the number of paths associated with the effect's source layer.</p> <pre>PF_NumPaths(PF_ProgPtr effect_ref, long *num_pathsPL);</pre>
PF_PathInfo	<p>Retrieves the PF_PathID for the specified path.</p> <pre>PF_PathInfo(PF_ProgPtr effect_ref, long indexL, PF_PathID *unique_idP);</pre>

TABLE 37: PF_PATHQUERYSUITE

Function	Purpose
PF_CheckoutPath	<p>Acquires the PF_PathOutlinePtr for the path at the specified time.</p> <pre>PF_CheckoutPath(PF_ProgPtr effect_ref, PF_PathID unique_id, long what_time, long time_step, unsigned long time_scale, PF_PathOutlinePtr *pathPP);</pre>
PF_CheckinPath	<p>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.</p> <pre>PF_CheckinPath(PF_ProgPtr effect_ref, PF_PathID unique_id, PF_Boolean changedB, PF_PathOutlinePtr pathP);</pre>

ACCESSING CAMERA AND LIGHT INFORMATION

Using functions provided in the [AEGP_PFInterfaceSuite](#), 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 38: PIXEL TYPES FOR DIFFERENT COLOR SPACES

Pixel Type	Data Structure
8 bpc ARGB	<pre>typedef struct { unsigned char alpha, red, green, blue; } PF_Pixel8;</pre>
16 bpc ARGB	<pre>typedef struct { unsigned short alpha, red, green, blue; } PF_Pixel16;</pre>
32 bpc ARGB	<pre>typedef struct { PF_FpShort alpha, red, green, blue; } PF_PixelFloat, PF_Pixel32;</pre>
HLS (Hue, Lightness, Saturation)	<pre>typedef PF_Fixed PF_HLS_PIXEL[3]</pre>
YIQ (luminance, in-phase chrominance, quadrature chrominance)	<pre>typedef PF_Fixed PF_YIQ_PIXEL[3]</pre>

Plug-ins can draw on image processing algorithms written for nearly any color space by using the following callback functions.

TABLE 39: 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 <code>PF_RGB_TO_HLS</code> .
HLStoRGB	Given an HLS pixel, returns an RGB pixel. Replaces <code>PF_HLS_TO_RGB</code> .
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 <code>PF_RGB_TO_YIQ</code> .
YIQtoRGB	Given a YIQ pixel, returns an RGB pixel. Replaces <code>PF_YIQ_TO_RGB</code> .

TABLE 39: 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, returns 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 999. 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`](#).

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\(\)`](#) and [`PF_PROGRESS\(\)`](#). 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?

Deeeeeeep, 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](#)) 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](#), [PF_OutFlag_WIDE_TIME_INPUT](#), [PF_OutFlag_I_USE_SHUTTER_ANGLE](#), [PF_OutFlag_I_SYNTHESIZE_AUDIO](#), [PF_OutFlag2_I_USE_3D_CAMERA](#), and [PF_OutFlag2_I_USE_3D_LIGHTS](#) all influence caching decisions.

Supporting [dynamic outflags](#) 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?

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`](#) 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

Starting in After Effects 7.0, the SmartFX API provides bidirectional communication between effects and After Effects, enabling many performance optimizations and providing previously unavailable dependency information.

Normal effect plug-ins are given a full-sized input buffer, and asked to render a full-sized output buffer. While output [extent_hint](#) 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.

WHAT'S NEW IN CS3 (8.0)?

The API has not changed; however, we've fixed a couple of cache invalidation and rect determination bugs, so that SmartFX will behave as documented.

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`](#) will receive the SmartFX calls [`PF_Cmd_SMART_PRE_RENDER`](#) and [`PF_Cmd_SMART_RENDER`](#), instead of the older [`PF_Cmd_FRAME_SETUP`](#) / [`PF_Cmd_RENDER`](#) / [`PF_Cmd_FRAME_SETDOWN`](#) 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](#), any non-layer parameters needed must be retrieved using [PF_CHECKOUT_PARAM](#)

TABLE 40: PF_PRENDEREXTRA

Member	Purpose
PF_PreRenderInput	<p>Describes what After Effects needs rendered (in the PF_RenderRequest), and the bit depth requested (in the aptly-named bitdepth member).</p> <pre>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;</pre> <p>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.</p> <p>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:</p> <pre>PF_ChannelMask_ALPHA PF_ChannelMask_RED PF_ChannelMask_GREEN PF_ChannelMask_BLUE PF_ChannelMask_ARGB</pre> <p>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, which tells After Effects that the effect may set alpha to non-zero values for such pixels, restoring them to visibility.</p>

TABLE 40: PF_PRERENDEREXTRA

Member	Purpose
PF_PreRenderOutput	<p>Filled in by the effect to tell After Effects what output it plans to generate, based on the input.</p> <pre>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> <p>pre_render_data will be passed back to the effect during PF_Cmd SMART RENDER.</p> <p>Currently, the only PF_RenderOutputFlags is PF_RenderOutputFlag_RETURNS_EXTRA_PIXELS.</p>

TABLE 40: PF_PRERENDEREXTRA

Member	Purpose
PF_PreRenderCallbacks	<p>Currently, there is only one callback - <code>checkout_layer</code>. <code>checkout_idL</code> is chosen by the effect. It must be positive and unique. After Effects populates the <code>PF_CheckoutResult</code>.</p> <pre> 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); </pre> <pre> 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; </pre> <p><code>result_rect</code> can be empty. <code>max_result_rect</code> is the largest the output could possibly be, if the host asked for all of it. If <code>solid</code> is <code>TRUE</code>, the entire <code>result_rect</code> has opaque alpha.</p> <p><code>ref_width</code> and <code>ref_height</code> 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.</p>

TABLE 41: PF_PRERENDEROUTPUT

Member	Purpose
<code>result_rect</code>	The output (in layer coordinates) resulting from the render request (can be empty). This cannot be bigger than the input request rectangle (unless <code>PF_RenderOutputFlag_RETURNS_EXTRA_PIXELS</code> is set), but can be smaller.
<code>max_result_rect</code>	The maximum size the output could possibly be, if After Effects requested all of it. This must not vary depending on requested output size.
<code>solid</code>	Set this <code>TRUE</code> if every pixel in the output will be fully opaque. Set if possible; it enables certain optimizations.

TABLE 41: PF_PRERENDEROUTPUT

Member	Purpose
reserved	Ignore.
flags	Currently, the only flag is <code>PF_RenderOutputFlag_RETURNS_EXTRA_PIXELS</code> , 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 <code>out_data>frame_data</code> , as with regular (non-smart) effects. Since PF_Cmd_SMART_PRE_RENDER can be called with no corresponding PF_Cmd_SMART_RENDER , effects must never delete this data themselves; once the effect returns from PF_Cmd_SMART_PRE_RENDER , After Effects owns this data and will dispose of it (using either the following function or a standard <code>free</code> call).
delete_pre_render_data_func	Point this to a function that will eventually be called to delete the <code>pre_render_data</code> .

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 as 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](#) 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 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 42: PF_SMARTRENDEREXTRA

Member	Purpose
<code>PF_SmartRenderInput</code>	Consists of a PF_RenderRequest , the bitdepth, and a pointer to <code>pre_render_data</code> (allocated during <code>PF_Cmd_SMART_PRE_RENDER</code>). This <code>PF_SmartRenderInput</code> is identical to that passed in the corresponding <code>PF_Cmd_SMART_PRE_RENDER</code> .

TABLE 42: PF_SMARTRENDEREXTRA

Member	Purpose
PF_SmartRenderCallbacks	<pre>PF_Err checkout_layer_pixels(PF_ProgPtr effect_ref, A_long checkout_idL, PF_EffectWorld **pixels);</pre> <p>This is used to actually access the pixels in layers checked out during <i>PF_Cmd_SMART_PRE_RENDER</i>. The returned <i>PF_EffectWorld</i> is valid for duration of current command or until checked in.</p> <p>You are only allowed to call <i>checkout_layer_pixels</i> only once with the <i>checkout_idL</i> used earlier in <i>PF_Cmd_SMART_PRERENDER</i>. There must be a one-to-one mapping between the number of checkouts made in <i>PF_Cmd_SMART_PRERENDER</i> and <i>PF_Cmd_SMART_RENDER</i>. To call <i>checkout_layer_pixels</i> more than once on a layer, you should call checkout_layer on the same layer again with a different unique <i>checkout_idL</i> in <i>PF_Cmd_SMART_PRERENDER</i> and then use that <i>checkout_idL</i> to do another <i>checkout_layer_pixels</i> in <i>PF_Cmd_SMART_RENDER</i>.</p> <pre>PF_Err checkin_layer_pixels(PF_ProgPtr effect_ref, A_long checkout_idL);</pre> <p>It isn't necessary to call (After Effects cleans up all such checkouts when the effect returns from <i>PF_Cmd_SMART_RENDER</i>), but useful to free up memory.</p> <pre>PF_Err checkout_output(PF_ProgPtr effect_ref, PF_EffectWorld **output);</pre> <p>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).</p> <p>NOTE: For optimal memory usage, request the output as late as possible, and request inputs as few at a time as possible.</p>

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](#). 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](#) and do not subsequently check it out in [PF_Cmd_SMART_RENDER](#), 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](#) during [PF_Cmd_GLOBAL_SETUP](#), 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](#). Parameters that have a custom UI should set [PF_PUI_CONTROL](#) when [adding the parameter](#).

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](#)->e_type and the various events are described below.

TABLE 43: EVENTS

Event	Indicates
<i>PF_Event_NEW_CONTEXT</i>	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 contains valid context and type, but everything else should be ignored.
<i>PF_Event_ACTIVATE</i>	The user activated a new context (probably by bringing a window into the foreground). PF_EventUnion is empty.
<i>PF_Event_DO_CLICK</i>	The user clicked within the effect's UI. PF_EventUnion 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 <i>not</i> block until mouse-up; instead, rely on PF_Event_DRAG .

TABLE 43: EVENTS

Event	Indicates
<i>PF_Event_DRAG</i>	Also a Click Event, <i>PF_EventUnion</i> contains a <i>PF_DoClickEventInfo</i> . Request this by returning <code>send_drag == TRUE</code> from <i>PF_Event_DO_CLICK</i> . Do this so After Effects can see new data from the user's changes.
<i>PF_Event_DRAW</i>	Draw! <i>PF_EventUnion</i> contains a <i>PF_DrawEventInfo</i> .
<i>PF_Event_DEACTIVATE</i>	The user has deactivated a context (probably by bringing another window into the foreground). <i>PF_EventUnion</i> is empty.
<i>PF_Event_CLOSE_CONTEXT</i>	A context has been closed by the user. <i>PF_EventUnion</i> will be empty.
<i>PF_Event_IDLE</i>	A context is open but nothing is happening. <i>PF_EventUnion</i> is empty.
<i>PF_Event_KEYDOWN_OBSOLETE</i>	As of 7.0, this old message is no longer sent, though it remains in the enumeration.
<i>PF_Event_ADJUST_CURSOR</i>	The mouse is over the plug-in's UI. Set the cursor by changing the <i>PF_CursorType</i> in the <i>PF_AdjustCursorEventInfo</i> . Use OS-specific calls to implement a custom cursor; tell After Effects you've done so by setting <i>PF_CursorType</i> to <i>PF_Cursor_CUSTOM</i> . Use an After Effects cursor whenever possible to preserve interface continuity.
<i>PF_Event_KEYDOWN</i>	New in 7.0. Keystroke. <i>PF_EventUnion</i> contains a <i>PF_KeyDownEvent</i> .

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](#). The *PF_EventUnion* (sent in the *PF_EventExtra*) varies with the event type, and contains information specific to that event.

TABLE 44: PF_EVENTEXTRA

Member	Purpose
<code>contextH</code>	Handle to the PF_Context . This drawing context is used with the Drawbot suites for drawing, and also for the UI callbacks .
<code>e_type</code>	Which event is occurring.

TABLE 44: PF_EVENTEXTRA

Member	Purpose
u	A PF_EventUnion containing information specific to the event.
effect_win	<p>A PF_EffectWindowInfo about the event if it occurs within the effects window.</p> <p>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.</p>
cbs	Pointer to UI callbacks , 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	<p>One or more of the following, combined with a bitwise OR operation:</p> <p>PF_EO_NONE</p> <p>PF_EO_HANDLED_EVENT tells After Effects you've handled the event.</p> <p>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.</p> <p>PF_EO_NEVER_UPDATE prevents After Effects from rerendering the composite until the user stops clicking and dragging.</p> <p>PF_EO_UPDATE_NOW tells After Effects to update the view immediately after the event returns after calling PF_InvalidateRect</p>

PF_CONTEXT

PF_Context details the event's UI context.

TABLE 45: 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 suites.
*reserved_paneP	Do not change.

If an event occurs in the ECP, an PF_EffectWindowInfo is sent in PF_EventExtra.

TABLE 46: 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 47: PF_DoCLICKEVENTINFO

Member	Purpose
<code>when</code>	The (OS-level) time at which the click occurred.
<code>screen_point</code>	Where, in screen coordinates, the click occurred. For Custom Comp UI, these coordinates can be converted to composition coordinates using the UI Callbacks . See the CCU sample project for an example.
<code>num_clicks</code>	The number of clicks that occurred.
<code>modifiers</code>	Which modifier keys (if any) were held down during click.
<code>continue_refcon[4]</code>	An array of 4 <code>A_intptr_t</code> the plug-in can use to store information during a click-drag-drag sequence.
<code>send_drag</code>	Set this flag to <code>TRUE</code> to indicate continued dragging. The next click event will then effectively be a drag event.
<code>last_time</code>	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](#) (rather than the dimensions of your input layer, as you would during [PF_Cmd_RENDER](#)).

TABLE 48: PF_DRAWEVENTINFO

Member	Purpose
<code>update_rect</code>	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 . See the CCU sample project for an example.
<code>depth</code>	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 49: PF_KEYDOWNEVENT

Member	Purpose
<code>when</code>	Time at which the click occurred.
<code>screen_point</code>	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 . See the CCU sample project for an example.

TABLE 49: PF_KEYDOWNEVENT

Member	Purpose
key_code	<p>Either a character code (for printable characters, we use the unshifted upper case version; A not a, 7 not &), or a control code:</p> <p>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</p>
modifiers	<p>Which (if any) modifier keys were down during the key press.</p> <p>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</p>

ADJUSTCURSOR

The cursor has moved onto (but not off of) the effect's custom UI, to allow the effect to change the cursor.

TABLE 50: 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 Callbacks . 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 <code>PF_Cursor_CUSTOM</code> if you have set the cursor yourself using OS-specific calls. If you don't want to override the cursor, set this to <code>PF_Cursor_NONE</code> , or simply ignore this message.

ARBITRARY PARAMETERS EVENT

After Effects needs your plug-in to manage its arbitrary data parameter(s). Though arbitrary data types are not required for custom UI support, `PF_ArbParamsExtra` follows the `EventInfo` model.

TABLE 51: PF_ARBPARAMSEXTRA

Member	Purpose
which_function	A <code>PF_FunctionSelector</code> indicating which function is called
id	Used by After Effects; will match the ID assigned to the arbitrary data type during <code>PF_Cmd_PARAM_SETUP</code> .
padding	Used for byte-alignment

TABLE 51: PF_ARBPARAMSEXTRA

Member	Purpose
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)

CUSTOM UI AND DRAWBOT

New in CS5, custom UI has moved from a non-composited drawing model to 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 now only occur during PF_Event_DRAW, and not during PF_Event_DRAG or PF_Event_DO_CLICK, as was allowed in CS4 and earlier. To use Drawbot, first get the drawing reference by passing in PF_Context to a new suite call [PF_GetDrawingReference](#). If a non-NULL drawing reference is returned, use it to get the supplier and surface references from [DRAWBOT_DrawbotSuite](#).

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](#) to match the host application UI. Your users will thank you.

REDRAWING

New in CS5, we now allow the effect to specify a redraw with much finer granularity. In order to redraw a specific area of a pane, we recommend the following:

- 1) Call [PF_InvalidRect](#) 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.
- 2) Set the [event outflag](#) 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](#), which will cause a redraw of the entire ECW, comp, and layer windows.

PF_EFFECTCUSTOMUISUITE

Enables an effect to get the drawing reference. This is the first call needed to use Drawbot.

TABLE 52: PF_EFFECTCUSTOMUISUITE1

Function	Purpose
PF_GetDrawingReference	Get the drawing reference. <pre>PF_GetDrawingReference(const PF_ContextH effect_contextH, DRAWBOT_DrawRef *referenceP);</pre>

DRAWBOT_DRAWBOTSUITE

Using the Drawbot reference, get the supplier and surface references.

TABLE 53: DRAWBOT_DRAWBOTSUITE1

Function	Purpose
GetSupplier	Get the supplier reference. Needed to use DRAWBOT_SupplierSuite . <pre>GetSupplier(DRAWBOT_DrawRef in_drawbot_ref, DRAWBOT_SupplierRef *out_supplierP);</pre>
GetSurface	Get the surface reference. Needed to use DRAWBOT_SurfaceSuite . <pre>GetSurface(DRAWBOT_DrawRef in_drawbot_ref, DRAWBOT_SurfaceRef *out_surfaceP);</pre>

DRAWBOT_SUPPLIERSUITE

Calls to create and release drawing tools, get default settings, and query drawing capabilities.

TABLE 54: DRAWBOT_SUPPLIERSUITE1

Function	Purpose
NewPen	Create a new pen. Release this using ReleaseObject . <pre>NewPen(DRAWBOT_SupplierRef in_supplier_ref, const DRAWBOT_ColorRGBA *in_colorP, float in_size, DRAWBOT_PenRef *out_penP);</pre>
NewBrush	Create a new brush. Release this using ReleaseObject . <pre>NewBrush(DRAWBOT_SupplierRef in_supplier_ref, const DRAWBOT_ColorRGBA *in_colorP, DRAWBOT_BrushRef *out_brushP);</pre>
SupportsText	Check if current supplier supports text. <pre>SupportsText(DRAWBOT_SupplierRef in_supplier_ref, DRAWBOT_Boolean *out_supports_textB);</pre>
GetDefaultFontSize	Get the default font size. <pre>GetDefaultFontSize(DRAWBOT_SupplierRef in_supplier_ref, float *out_font_sizeF);</pre>
NewDefaultFont	Create a new font with default settings. You can pass the default font size from GetDefaultFontSize. Release this using ReleaseObject . <pre>NewDefaultFont(DRAWBOT_SupplierRef in_supplier_ref, float in_font_sizeF, DRAWBOT_FontRef *out_fontP);</pre>

TABLE 54: DRAWBOT_SUPPLIERSUITE1

Function	Purpose
NewImageFromBuffer	<p>Create a new image from buffer passed to in_dataP. Release this using ReleaseObject.</p> <pre> 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); </pre> <p>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</p>
NewPath	<p>Create a new path. Release this using ReleaseObject.</p> <pre> NewPath(DRAWBOT_SupplierRef in_supplier_ref, DRAWBOT_PathRef *out_pathP); </pre>
SupportsPixelLayoutBGRA	<p>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).</p> <pre> SupportsPixelLayoutBGRA(DRAWBOT_SupplierRef in_supplier_ref, DRAWBOT_Boolean *out_supports_bgraPB); </pre>
PrefersPixelLayoutBGRA	<pre> PrefersPixelLayoutBGRA(DRAWBOT_SupplierRef in_supplier_ref, DRAWBOT_Boolean *out_prefers_bgraPB); </pre>
SupportsPixelLayoutARGB	<pre> SupportsPixelLayoutARGB(DRAWBOT_SupplierRef in_supplier_ref, DRAWBOT_Boolean *out_supports_argbPB); </pre>
PrefersPixelLayoutARGB	<pre> PrefersPixelLayoutARGB(DRAWBOT_SupplierRef in_supplier_ref, DRAWBOT_Boolean *out_prefers_argbPB); </pre>

TABLE 54: DRAWBOT_SUPPLIERSUITE1

Function	Purpose
RetainObject	<p>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.</p> <pre>RetainObject(DRAWBOT_ObjectRef in_obj_ref);</pre>
ReleaseObject	<p>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.</p> <pre>ReleaseObject(DRAWBOT_ObjectRef in_obj_ref);</pre>

DRAWBOT_SURFACESUITE

Calls to draw on the surface, and to query and set drawing settings.

TABLE 55: DRAWBOT_SURFACESUITE1

Function	Purpose
PushStateStack	<p>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.</p> <pre>PushStateStack(DRAWBOT_SurfaceRef in_surface_ref);</pre>
PopStateStack	<p>Pop the last pushed surface state off the stack.</p> <pre>PopStateStack(DRAWBOT_SurfaceRef in_surface_ref);</pre>
PaintRect	<p>Paint a rectangle with a color on the surface.</p> <pre>PaintRect(DRAWBOT_SurfaceRef in_surface_ref, const DRAWBOT_ColorRGBA *in_colorP, const DRAWBOT_RectF32 *in_rectPR);</pre>

TABLE 55: DRAWBOT_SURFACE_SUITE1

Function	Purpose
FillPath	<p>Fill a path using a brush and fill type.</p> <pre>FillPath(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_BrushRef in_brush_ref, DRAWBOT_PathRef in_path_ref, DRAWBOT_FillType in_fill_type);</pre> <p>DRAWBOT_FillType is one of the following: kDRAWBOT_FillType_EvenOdd, kDRAWBOT_FillType_Winding</p>
StrokePath	<p>Stroke a path using a pen.</p> <pre>StrokePath(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_PenRef in_pen_ref, DRAWBOT_PathRef in_path_ref);</pre>
Clip	<p>Clip the surface.</p> <pre>Clip(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_SupplierRef in_supplier_ref, const DRAWBOT_Rect32 *in_rectPR);</pre>
GetClipBounds	<p>Get clip bounds.</p> <pre>GetClipBounds(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_Rect32 *out_rectPR);</pre>
IsWithinClipBounds	<p>Checks whether a rect is within the clip bounds.</p> <pre>IsWithinClipBounds(DRAWBOT_SurfaceRef in_surface_ref, const DRAWBOT_Rect32 *in_rectPR, DRAWBOT_Boolean *out_withinPB);</pre>
Transform	<p>Transform the last surface state.</p> <pre>Transform(DRAWBOT_SurfaceRef in_surface_ref, const DRAWBOT_MatrixF32 *in_matrixP);</pre>

TABLE 55: DRAWBOT_SURFACESUITE1

Function	Purpose
DrawString	<p>Draw a string.</p> <pre> 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); </pre> <p>DRAWBOT_TextAlignment is one of the following: kDRAWBOT_TextAlignment_Left, kDRAWBOT_TextAlignment_Center, kDRAWBOT_TextAlignment_Right</p> <p>DRAWBOT_TextTruncation is one of the following: kDRAWBOT_TextTruncation_None, kDRAWBOT_TextTruncation_End, kDRAWBOT_TextTruncation_EndEllipsis, kDRAWBOT_TextTruncation_PathEllipsis</p>
DrawImage	<p>Draw an image created using NewImageFromBuffer() on the surface. Alpha = [0.0f, 1.0f].</p> <pre> DrawImage(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_ImageRef in_image_ref, const DRAWBOT_PointF32 *in_originP, float in_alpha); </pre>
SetInterpolationPolicy	<pre> SetInterpolationPolicy(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_InterpolationPolicy in_interp); </pre> <p>DRAWBOT_InterpolationPolicy is one of the following: kDRAWBOT_InterpolationPolicy_None, kDRAWBOT_InterpolationPolicy_Med, kDRAWBOT_InterpolationPolicy_High</p>
GetInterpolationPolicy	<pre> GetInterpolationPolicy(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_InterpolationPolicy *out_interpP); </pre>

TABLE 55: DRAWBOT_SURFACESUITE1

Function	Purpose
SetAntiAliasPolicy	<pre>SetAntiAliasPolicy(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_AntiAliasPolicy in_policy);</pre> <p>DRAWBOT_AntiAliasPolicy is one of the following: kDRAWBOT_AntiAliasPolicy_None, kDRAWBOT_AntiAliasPolicy_Med, kDRAWBOT_AntiAliasPolicy_High</p>
GetAntiAliasPolicy	<pre>GetAntiAliasPolicy(DRAWBOT_SurfaceRef in_surface_ref, DRAWBOT_AntiAliasPolicy *out_policyP);</pre>
Flush	<p>Flush drawing. This is not always needed, and if overused, may cause excessive redrawing and flashing.</p> <pre>Flush(DRAWBOT_SurfaceRef in_surface_ref);</pre>

DRAWBOT_PATHSUITE

Calls to draw paths.

TABLE 56: DRAWBOT_PATHSUITE1

Function	Purpose
MoveTo	<p>Move to a point.</p> <pre>MoveTo(DRAWBOT_PathRef in_path_ref, float in_x, float in_y);</pre>
LineTo	<p>Add a line to the path.</p> <pre>LineTo(DRAWBOT_PathRef in_path_ref, float in_x, float in_y);</pre>
BezierTo	<p>Add a cubic bezier to the path.</p> <pre>BezierTo(DRAWBOT_PathRef in_path_ref, const DRAWBOT_PointF32 *in_pt1P, const DRAWBOT_PointF32 *in_pt2P, const DRAWBOT_PointF32 *in_pt3P);</pre>

TABLE 56: DRAWBOT_PATHSUITE1

Function	Purpose
AddRect	Add a rect to the path. <pre>AddRect(DRAWBOT_PathRef in_path_ref, const DRAWBOT_RectF32 *in_rectPR);</pre>
AddArc	Add a arc to the path. Zero start degrees == 3 o'clock. Sweep is clockwise. Units for angle are in degrees. <pre>AddArc(DRAWBOT_PathRef in_path_ref, const DRAWBOT_PointF32 *in_centerP, float in_radius, float in_start_angle, float in_sweep);</pre>
Close	Close the path. <pre>Close(DRAWBOT_PathRef in_path_ref);</pre>

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 57: PF_EFFECTCUSTOMUIOVERLAYTHEMESUITE1

Function	Purpose
PF_GetPreferredForegroundColor	Get the preferred foreground color. <pre>PF_GetPreferredForegroundColor(DRAWBOT_ColorRGBA *foreground_colorP);</pre>
PF_GetPreferredShadowColor	Get the preferred shadow color. <pre>PF_GetPreferredShadowColor(DRAWBOT_ColorRGBA *shadow_colorP);</pre>
PF_GetPreferredStrokeWidth	Get the preferred foreground & shadow stroke width. <pre>PF_GetPreferredStrokeWidth(float *stroke_widthPF);</pre>

TABLE 57: PF_EFFECTCUSTOMUIOVERLAYTHEMESUITE1

Function	Purpose
PF_GetPreferredVertexSize	Get the preferred vertex size. <pre>PF_GetPreferredVertexSize(float *vertex_sizePF);</pre>
PF_GetPreferredShadowOffset	Get the preferred shadow offset. <pre>PF_GetPreferredShadowOffset(A_LPoint *shadow_offsetP);</pre>
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. <pre>PF_StrokePath(const DRAWBOT_DrawRef drawbot_ref, const DRAWBOT_PathRef path_ref PF_Boolean draw_shadowB);</pre>
PF_FillPath	Fills the path with overlay theme foreground color. Optionally draw the shadow using the overlay theme shadow color. <pre>PF_FillPath(const DRAWBOT_DrawRef drawbot_ref, const DRAWBOT_PathRef path_ref PF_Boolean draw_shadowB);</pre>
PF_FillVertex	Fills a square vertex around the center point using the overlay theme foreground color and vertex size. <pre>PF_FillVertex(const DRAWBOT_DrawRef drawbot_ref, const A_FloatPoint *center_pointP PF_Boolean draw_shadowB);</pre>

THE WAY CUSTOM UI USED TO BE

(Custom UI from 7.0 to CS4 (9.0) was handled differently, and the following is left in for the benefit of plug-ins that are backwards compatible.) We've completely overhauled our imaging subsystem to take advantage of GDI+ on Windows and Quartz Extreme on Mac OS. Internally, this meant we no longer paint directly to the screen, can't use any Quickdraw calls (Mac), and all Windows HWNDs must be created with a flag that precludes the use of many GDI calls, to preserve hardware acceleration. Additionally, it is only safe to paint when receiving a paint event from the operating system as the appropriate graphics context is not available at other times (i.e. during a mouse event).

WHAT THIS MEANS TO YOU

We've implemented a compatibility layer. The `cgrafptr` passed in all events now refers to an offscreen context. This means that, on Windows, if you use [PF_CGrafData_HDC](#), you can't call `WindowFromDC()` as there is no associated window. `GetMouse()` on Mac OS will now return global coordinates, as the current port is offscreen; `GlobalToLocal()` and `LocalToGlobal()` can no longer be used.

Any offscreen drawing during an event will not be immediately visible in After Effects. You will need to return [PF_OutFlag_REFRESH_UI](#) to make your changes visible.

If your plug-in returned `PF_EventExtra.u.do_click.send_drag` as `TRUE` from the [PF_Event_DO_CLICK](#) event, we will update from the offscreen buffer so the user can see it.

If you *must* implement a hard tracking loop without using `send_drag`, use [PF_GetMouse\(\)](#) and [PF_ForceCustomUIUpdate\(\)](#). We've added [PF_ConvertLocalToGlobal\(\)](#) to allow on-screen positioning of elements (like menus) during any of these events. Still, tracking loops using `send_drag` allow for more changes within After Effects (and fewer in your code) in the future.

Composition panel and layer panel custom user interface has grown more expensive. With 7.0, custom user interface is much more expensive.

HOW EXPENSIVE?

For custom composition and layer panel UIs, we maintain an offscreen buffer the size of the composition/layer panel whenever your plug-in is selected (in the ECP). For every draw event, we draw the entire composition panel into the offscreen, send the draw event to the plug-in, traverse all pixels of the offscreen to ensure they're opaque (Quickdraw/GDI calls can change the opacity), then blit it to the drawing surface. For updates while processing other events or immediately after an event, we confirm all pixels are opaque, invalidate, and force the OS to send us a paint message.

See? We *told* you it was expensive.

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` 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 58: UI CALLBACKS

Function	Purpose
LAYER_TO_COMP	Transforms layer panel coordinates to the composition panel coordinates. <pre>PF_Err LAYER_TO_COMP (long curr_time, long time_scale, PF_FixedPoint *pt);</pre>
COMP_TO_LAYER	Transforms composition panel coordinates to the layer panel coordinates. <pre>PF_Err COMP_TO_LAYER (long curr_time, long time_scale, PF_FixedPoint *pt);</pre>
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. <pre>PF_Err GET_COMP2LAYER_XFORM (long curr_time, long time_scale, long *exists, PF_FloatMatrix *comp2layer);</pre>

TABLE 58: UI CALLBACKS

Function	Purpose
GET_LAYER2COMP_XFORM	<p>Returns the transformation matrix used to convert from the layer panel to the composition panel. This always exists.</p> <pre>PF_Err GET_LAYER2COMP_XFORM (long curr_time, long time_scale, PF_FloatMatrix *layer2comp);</pre>
SOURCE_TO_FRAME	<p>Transforms the source coordinates in the current context to screen coordinates. Screen (frame) coordinates are affected by the current zoom level.</p> <pre>PF_Err SOURCE_TO_FRAME(PF_FixedPoint *pt);</pre>
FRAME_TO_SOURCE	<p>Transforms the screen coordinates identified by *pt to the source coordinates of the current context.</p> <pre>PF_Err FRAME_TO_SOURCE(PF_FixedPoint *pt);</pre>
INFO_GET_PORT	<p>Returns the Mac OS cgrafptr for the info window so you can draw whatever you want into it. If you're writing a plug-in for Windows, you'll need to convert the cgrafptr to an HDC, or utilize components of Apple's QuickTime for Windows SDK.</p> <pre>PF_Err INFO_GET_PORT(void **cgrafptr_addr);</pre>
PF_GET_PLATFORM_DATA	<p>Retrieves platform-specific data. For plug-ins loaded with localized resource files, PF_PlatformData_RES_FILE_PATH will point to the external file, not the plug-in file. Use PF_PlatformData_EXE_FILE_PATH if you want the path of your plug-in.</p> <pre>PF_Err PF_GET_PLATFORM_DATA (PF_PlatformDataID which, void *ppData);</pre> <p>PF_PlatformDataID can have the following values:</p> <pre>PF_PlatformData_MAIN_WND PF_PlatformData_EXE_FILE_PATH PF_PlatformData_RES_FILE_PATH PF_PlatformData_RES_REFNUM // Mac OS PF_PlatformData_RES_DLLINSTANCE // Win</pre>

TIPS AND TRICKS

UI PERFORMANCE

Experiment with [PF_EO_ALWAYS_UPDATE](#) and [PF_EO_NEVER_UPDATE](#), 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](#). Yes, this means the effect needs to [cheat](#) 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](#) for more details).

AUDIO DATA STRUCTURES

The following data types are used by After Effects to describe audio data.

TABLE 59: AUDIO DATA STRUCTURES

Structure	Description
<code>PF_SoundFormat</code>	Indicates whether the audio is in unsigned pulse code modulation (PCM), signed PCM, or floating point format.
<code>PF_SoundSampleSize</code>	Samples are in 1, 2, or 4 byte format.
<code>PF_SoundChannels</code>	Indicates whether the audio is mono or stereo.

TABLE 59: AUDIO DATA STRUCTURES

Structure	Description
<code>PF_SoundFormatInfo</code>	Contains the sampling rate, number of channels, sample size, and format of the audio to which it refers.
<code>PF_SoundWorld</code>	Use <code>PF_SoundWorlds</code> to represent audio. In addition to a <code>PF_SoundFormatInfo</code> , 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](#) to retrieve an audio layer. This layer is opaque; use [PF_GET_AUDIO_DATA](#) 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](#).

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?

WHAT'S NEW IN CS5?

The [AEGP entry point function](#) has changed, in that we no longer pass in the full file path and resource path. We plan to provide these paths as Unicode by adding a new call, `AEGP_GetPluginPaths`.

We have also revised the AEGP suites to use Unicode platform paths: [AEGP_ProjSuite](#), [AEGP_FootageSuite](#), [AEGP_OutputModuleSuite](#), [AEGP_FIMSuite](#), [AEGP_PersistentDataSuite](#), and [AEGP_RegisterSuite](#).

`AEGP_MENU_INSERT_AT_BOTTOM` and `AEGP_MENU_INSERT_AT_TOP` are no longer supported for [AEGP_InsertMenuCommand](#) in the `AEGP_Menu_WINDOW`. We recommend using `AEGP_MENU_INSERT_SORTED`.

We have also added a new footage mode that does not attempt to guess the alpha, in the updated [AEGP_NewFootage](#).

In the Layer Suite, we've added [AEGP_GetLayerLabel](#) and [AEGP_SetLayerLabel](#).

WHAT'S NEW IN CS4 (9.0)?

[AEGP_DynamicStreamSuite](#) has been updated to support the new Position keyframes that have been separated into individual XYZ properties.

[AEGP_ItemSuite](#), [AEGP_CompSuite](#), [AEGP_LayerSuite](#), [AEGP_StreamSuite](#), [AEGP_DynamicStreamSuite](#), [AEGP_TextDocumentSuite](#), [AEGP_MarkerSuite](#), and [AEGP_MaskSuite](#) have all been updated to support Unicode strings.

[AEGP_MarkerSuite](#) has been updated to support marker durations.

WHAT'S NEW IN CS3 (8.0)?

Color management has introduced the concept of color spaces, requiring updates to numerous function suites. Vector layers can now be created via suite functions. Changes to how marker data is handled caused us to change how stream values are represented; this change flows throughout the stream-related suites. The new [AEGP_MarkerSuite](#) provides more tools for manipulating marker data. We've also made some motion blur awareness available via the [AEGP_CompSuite](#).

The AEGP API has grown by leaps and bounds. Unfettered by the constraints of the render pipeline, AEGPs can control almost everything After Effects does. It doesn't matter whether a command is triggered by a user, or by a plug-in. By exposing our internal functionality as we do, we enable plug-ins to go beyond scriptability, to offer complete control of After Effects.

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](#).

DIFFERENT TASKS, SAME API

AEGPs work in the same manner, regardless of specialization. They can be simple, just adding one menu item 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. While in some cases it might be more efficient to simply modify the underlying structure, by maintaining the opacity of the data types we allow for changes to our implementation without making you recompile (and redistribute) your plug-ins.

TABLE 60: AEGP API DATA TYPES

Type	Describes
<code>AEGP_ProjectH</code>	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 7.0, there will only ever be one open project.
<code>AEGP_ItemH</code>	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 <code>AEGP_ItemHs</code> until more specificity is required.

TABLE 60: AEGP API DATA TYPES

Type	Describes
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_FootageH	An item that can be rendered. Folders and compositions are the only items that are not footage.
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 can all become layers. Layers may be defined over sub-intervals of the composition's time interval.
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_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_MaskOutlineValH	The specific points which constitute the mask. The points in a mask outline are ordered, and the mask need not be closed.
AEGP_StreamRefH	Any parameter stream attached to a layer, in a composition. See the description of AEGP_GetNewLayerStream for a full list of stream types.
AEGP_RenderLayerContextH	State information at the time of a render request, sent to an Artisan by After Effects.
AEGP_PersistentBlobH	A “blob” of data containing the current preferences.
AEGP_Collection2H	A set of selected items.
AEGP_SoundDataH	The audio settings used for a given layer.
AEGP_RenderReceiptH AEGP_FrameReceiptH	Used by Artisans when rendering.
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_WorldH	A frame of pixels.

TABLE 60: AEGP API DATA TYPES

Type	Describes
AEGP_RenderOptionsH	The settings associated with a render queue item.
AEGP_RQItemRefH	An item in the render queue.
AEGP_OutputModuleRefH	An output module, attached to a specific AEGP_RQItemRef in the render queue.
AEGP_StreamVal2	The actual value of a stream at a specified time.
AEGP_MarkerVal	The data associated with a given timeline marker.
AEGP_TextOutlinesH	A reference to all the paths that make up the outlines of a given text layer.

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](#), 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).

In CS3 (8.0) and CS4 (9.0), the major version is 16, and the minor version is 24.

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`](#).

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.

EXAMPLE: ADDING A MENU ITEM

During your entry point function, use

[`CommandSuite>AEGP_GetUniqueCommand\(\)`](#) to obtain a command ID from After Effects, for use with [`AEGP_InsertMenuCommand\(\)`](#). Use a different ID for each menu item you add.

Using [`AEGP_RegisterSuite`](#)'s [`AEGP_RegisterCommandHook\(\)`](#), tell After Effects which function to call when your menu item(s) are selected. The function you register using [`AEGP_RegisterUpdateMenuHook\(\)`](#) 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.

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.

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.

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\(\)](#). But since `SPBasicSuite` itself is not thread safe, you'll need to stash off the function pointer in the main thread.

ACCESSING STREAM VALUES

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](#). 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](#), 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](#) (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\(\)](#) 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](#). New in CS3 (8.0); marker data is now handled as a pointer, not a handle, in `AEGP_StreamVal2`.

```
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;

```

DIFFERENT STREAM TYPES REQUIRE DIFFERENT ACCESSORS

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.

LAYERS

[AEGP_GetLayerStreamValue](#) 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](#). Masks don't have streams like layers do; they get their own enumeration. Access their streams using [AEGP_GetNewMaskStream](#).

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 [GetNewEffectStreamByIndex](#).

DYNAMIC STREAMS

The [AEGP_DynamicStreamSuite](#) provides functions for accessing paint, text, and other dynamic stream types.

PIXELS

Worlds of pixels are represented using the opaque [AEGP_WorldH](#).

SOUNDS

Audio is represented using the `AEGP_SoundDataFormat`.

```
typedef 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
```

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 61: DATA TYPES REQUIRING DISPOSAL

Data Type	Disposal function
<code>AEGP_FootageH</code>	<code>AEGP_DisposeFootage</code>
<code>AEGP_WorldH</code>	<code>AEGP_Dispose</code> (in <code>AEGP_WorldSuite</code>) Or <code>AEGP_DisposeTexture</code> , if layer texture created using <code>AEGP_RenderTexture</code>)
<code>AEGP_RenderReceiptH</code>	<code>AEGP_DisposeRenderReceipt</code>
<code>AEGP_EffectRefH</code>	<code>AEGP_DisposeEffect</code>
<code>AEGP_MaskRefH</code>	<code>AEGP_DisposeMask</code>
<code>AEGP_Collection2H</code>	<code>AEGP_DisposeCollection</code>
<code>AEGP_RenderOptionsH</code>	<code>AEGP_Dispose</code> (in <code>AEGP_RenderOptionsSuite</code>)

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 62: AEGP SUITES

Suite	Description
Project Suite	Reads and modifies project data.
Item Suite	Manages items within a project or composition. Folders, Compositions, Solids, and Footage are all items.
Composition Suite	Manages (and creates) compositions in a project, and composition-specific items like solids.
Memory Suite	Manage memory resources. Use this suite! Whenever memory-related errors are encountered, After Effects can report errors for you.
Layer Suite	Provides information about the layers within a composition, and the relationship(s) between the source and layer times
Stream Suite	Used to access the values of a layer's keyframe properties.
Dynamic Stream Suite	Used to access the characteristics of dynamic streams associated with a layer.
Keyframe Suite	Used to access and manipulate all keyframe data.
Effect Suite	Provides access to the effects applied to a layer. Use Stream suites to obtain effect keyframe information, use AEGP_EffectCallGeneric() to communicate with effects that you setup ahead of time to respond to your AEGP.
Mask Suite	Provides access to retrieve information about a layer's masks.
Mask Outline Suite	Used in conjunction with Stream Suite , this suite provides detailed information about the path rendered to make a layer's mask.
Footage Suite	Manages footage.
Register Suite	Used in conjunction with the Command 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.
Command Suite	Manage your AEGP's menu items. Used in conjunction with the Register Suite .

TABLE 62: AEGP SUITES

Suite	Description
<u>Utility Suite</u>	Supplies error message handling, AEGP version checking and access to After Effects' undo stack.
<u>IO_InSuite</u>	These suites are used by AEIO AEGPs, to query and manipulate input and output specification handles.
<u>IO_OutSuite</u>	
<u>PF Interface Suite</u>	The functions in this suite, while technically part of the AEGP API, are for use by effects.
<u>File Import Manager Suite</u>	Registers AEGP file and project importers as part of After Effects' file handling.
<u>Persistent Data Suite</u>	Query and manage all persistent data (i.e., the preferences file). AEGPs can also add their own data to the prefs.
<u>Collection Suite</u>	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.
<u>AEGP_World Suite</u>	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.
<u>Render Suite</u>	Finally, a way to get render pixels (and samples) from within an AEGP.
<u>Render Queue Suite</u>	Add and remove items from the render queue.
<u>Render Queue Item Suite</u>	Query and modify items in the render queue.
<u>Render Options Suite</u>	Query and manage all items exposed in a render item's options dialog.
<u>Output Module Suite</u>	Query and modify the output modules attached to items in the render queue.
<u>Sound Data Suite</u>	Functions for managing and accessing sound data.
<u>Canvas Suite</u>	Used by artisans to get information about (and rendered pixels from) the layers within the composition it's being asked to render.
<u>Artisan Utility Suite</u>	Manages the interface between the artisan and After Effects, including functions to move between render and instance contexts, and manage global data specific to the artisan.
<u>Camera Suite</u>	Query and control cameras.
<u>Light Suite</u>	Query and control lights.
<u>Query Transform Suite</u>	Provides access to and information about the transforms required to correctly place layers within a composition.
<u>Composite Suite</u>	Exposes After Effects' compositing functionality, including transfer modes, track matting, and good old fashioned bit copying.

TABLE 62: AEGP SUITES

Suite	Description
AEGP Iterate Suite	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

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 63: AEGP_FIMSUIE3

Function	Purpose
AEGP_RegisterImportFlavor	Registers the name of the file type(s) supported by the plug-in. Upon return, <code>imp_refP</code> will be a valid opaque reference, or <code>AE_FIM_ImportFlavorRef_NONE</code> . <pre>AEGP_RegisterImportFlavor(const char *nameZ, AE_FIM_ImportFlavorRef *imp_refP);</pre>
AEGP_RegisterImportFlavorFileTypes	Registers an array of file types and file extensions (the two arrays need not be of equal length) supported by the AEGP. <pre>AEGP_RegisterImportFlavorFileTypes(AE_FIM_ImportFlavorRef imp_ref, long num_filekindsL, const AEIO_FileKind *kindsAP, long num_fileextsL, const AEIO_FileKind *extsAP);</pre>
AEGP_RegisterImportFlavorImportCallbacks	Register the AEGP functions which will respond to import of different filetypes. <pre>AEGP_RegisterImportFlavorImportCallbacks(AE_FIM_ImportFlavorRef ref, AE_FIM_ImportFlags single_flag, const AE_FIM_ImportCallbacks *imp_cbsP);</pre>
AEGP_SetImportedItem	Designates an item as having been imported (possibly replacing an existing item), and sets associated import options. <pre>AEGP_SetImportedItem(AE_FIM_ImportOptions imp_options, AEGP_ItemH imported_itemH);</pre>

MANAGE PROJECTS

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](#). Use caution: the functions for opening and creating projects do not save changes to the project open when they are called!

TABLE 64: AEGP_PROJSUITE5

Function	Purpose
<code>AEGP_NumProjects</code>	Currently will never return more than 1. After Effects can have only one project open at a time. <code>AEGP_GetNumProjects()</code> <code>A_long</code> <code>*num_projPL</code>
<code>AEGP_GetIndProject</code>	Retrieves the current project. <code>AEGP_GetIndProject()</code> <code>A_long</code> <code>proj_indexL,</code> <code>AEGP_ProjectH</code> <code>*projPH</code> ;
<code>AEGP_GetProjectName</code>	Get the project name (up to <code>AEGP_MAX_PROJ_NAME_LEN + 1</code>) in length. <code>AEGP_GetProjectName()</code> <code>AEGP_ProjectH</code> <code>projH,</code> <code>A_char</code> <code>*nameZ</code> ;
<code>AEGP_GetProjectPath</code>	Get the path of the project (empty string the project hasn't been saved yet). The path is a handle to a NULL-terminated <code>A_UTF16Char</code> string, and must be disposed with <code>AEGP_FreeMemHandle</code> . <code>AEGP_GetProjectPath()</code> <code>AEGP_ProjectH</code> <code>projH,</code> <code>AEGP_MemHandle</code> <code>*unicode_pathPH</code>
<code>AEGP_GetProjectRootFolder</code>	Get the root of the project, which After Effects also treats as a folder. <code>AEGP_GetProjectRootFolder()</code> <code>AEGP_ProjectH</code> <code>projH,</code> <code>AEGP_ItemH</code> <code>*root_folderPH</code>
<code>AEGP_SaveProjectToPath</code>	Saves the entire project to the specified full path. The file path is a NULL-terminated UTF-16 string with platform separators. <code>AEGP_SaveProjectToPath()</code> <code>AEGP_ProjectH</code> <code>projH,</code> <code>const A_UTF16Char</code> <code>*pathZ</code> ;

TABLE 64: AEGP_PROJSUITE5

Function	Purpose
<code>AEGP_GetProjectTimeDisplay</code>	Retrieves the current time display settings. <pre>AEGP_GetProjectTimeDisplay(AEGP_ProjectH projH, AEGP_TimeDisplay *time_displayP);</pre>
<code>AEGP_SetProjectTimeDisplay</code>	Specifies the settings to be used for displaying time. <pre>AEGP_SetProjectTimeDisplay(AEGP_ProjectH projH, AEGP_TimeDisplay *time_displayP);</pre>
<code>AEGP_ProjectIsDirty</code>	Returns TRUE if the project has been modified since it was opened. <pre>AEGP_ProjectIsDirty(AEGP_ProjectH projH, A_Boolean *is_dirtyPB);</pre>
<code>AEGP_SaveProjectAs</code>	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. <pre>AEGP_SaveProjectAs(AEGP_ProjectH projH, const A_UTF16Char *pathZ);</pre>
<code>AEGP_NewProject</code>	Creates a new project. NOTE: Will close the current project without saving it first! <pre>AEGP_NewProject(AEGP_ProjectH *new_projectPH);</pre>
<code>AEGP_OpenProjectFromPath</code>	Opens a project from the supplied path, and returns its <code>AEGP_ProjectH</code> . The file path is a NULL-terminated UTF-16 string with platform separators. NOTE: Will close the current project without saving it first! <pre>AEGP_OpenProjectFromPath(const A_UTF16Char *pathZ, AEGP_ProjectH *projectPH);</pre>
<code>AEGP_GetProjectBitDepth</code>	Retrieves the project bit depth. <pre>AEGP_GetProjectBitDepth(AEGP_Projec tH projectH, AEGP_ProjBitDepth *bit_depthP);</pre> <p><code>AEGP_ProjBitDepth</code> will be one of the following:</p> <pre>AEGP_ProjBitDepth_8 AEGP_ProjBitDepth_16 AEGP_ProjBitDepth_32 (new in 7.0)</pre>

TABLE 64: AEGP_PROJSUITE5

Function	Purpose
<code>AEGP_SetProjectBitDepth</code>	Sets the project bit depth. Undoable. <pre>AEGP_SetProjectBitDepth(AEGP_ProjectH projectH, AEGP_ProjBitDepth bit_depth);</pre>

TABLE 65: AEGP_TIMEDISPLAY2

Member	Description
<i>Note: values in unused fields persist when After Effects is using a different display type.</i>	
<code>AEGP_TimeDisplayType type;</code>	One of the following: <code>AEGP_TimeDisplayType_TIMECODE</code> <code>AEGP_TimeDisplayType_FRAMES</code> <code>AEGP_TimeDisplayType_FEET_AND_FRAMES</code>
<code>A_char timebaseC;</code>	0 - 100. Only used for <code>AEGP_TimeDisplayType_TIMECODE</code> .
<code>A_Boolean non_drop_30B;</code>	When the timebase is 30 and the item's framerate is 29.97, determines whether to display as non-drop frame.
<code>A_char frames_per_footC;</code>	Only used for <code>AEGP_TimeDisplayType_FEET_AND_FRAMES</code> .
<code>A_long starting_frameL;</code>	Usually 0 or 1. Not used when type is usually 0 or 1, not used for <code>AEGP_TimeDisplayType_TIMECODE</code> .
<code>A_Boolean auto_timecode_baseB;</code>	New in 7.0. 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`. 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 66: AEGP_ITEMSUITE8

Function	Purpose
AEGP_GetFirstProjItem	<p>New in 7.0. Retrieves the first item in a given project.</p> <pre>AEGP_GetFirstProjItem(AEGP_ProjectH projectH, AEGP_ItemH *itemPH);</pre>
AEGP_GetNextProjItem	<p>Retrieves the next project item; *next_itemPH will be NULL after the last item. As of 7.0, this function call now takes an AEGP_ProjectH argument.</p> <pre>AEGP_GetNextProjItem(AEGP_ProjectH projectH, AEGP_ItemH itemH, AEGP_ItemH *next_itemPH);</pre>
AEGP_GetActiveItem	<p>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.</p> <pre>AEGP_GetActiveItem(AEGP_ItemH *itemPH,</pre>
AEGP_IsItemSelected	<p>Returns true if the Project window is active and the item is selected.</p> <pre>AEGP_IsItemSelected(AEGP_ItemH itemH, A_Boolean *selectedPB)</pre>
AEGP_SelectItem	<p>Toggles the selection state of the item, and (depending on deselect_othersB) can deselect others items.</p> <pre>AEGP_SelectItem(AEGP_ItemH itemH, A_Boolean selectB, A_Boolean deselect_othersB);</pre>

TABLE 66: AEGP_ITEMSUITE8

Function	Purpose
AEGP_GetItemType	<p>Gets type of an item. Note: solids don't appear in the project, but can be the source to a layer.</p> <pre>AEGP_GetItemType(AEGP_ItemH itemH, AEGP_ItemType *item_typeP);</pre> <p>Items are one of the following types:</p> <pre>AEGP_ItemType_NONE AEGP_ItemType_FOLDER AEGP_ItemType_COMP AEGP_ItemType_SOLID AEGP_ItemType_FOOTAGE</pre>
AEGP_GetTypeName	<p>Get name of type. (name length up to AEGP_MAX_TYPE_NAME_LEN + 1).</p> <pre>AEGP_GetTypeName(AEGP_ItemType item_type, A_char *nameZ);</pre>
AEGP_GetItemName	<p>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 .</p> <pre>AEGP_GetItemName(AEGP_PluginID pluginID, AEGP_ItemH itemH, AEGP_MemHandle *unicode_namePH);</pre>
AEGP_SetItemName	<p>New in 7.0. Specifies the name of the AEGP_ItemH. (name length has no limit). Undoable.</p> <pre>AEGP_SetItemName(AEGP_ItemH itemH, const A_UTF16Char *nameZ);</pre>
AEGP_GetItemID	<p>Returns the item's unique ID, which persists across saves and loads of the project.</p> <pre>AEGP_GetItemID(AEGP_ItemH itemH, A_long *item_idPL);</pre>

TABLE 66: AEGP_ITEMSUITE8

Function	Purpose
AEGP_GetItemFlags	<p>Get properties of an item. AEGP_GetItemFlags (</p> <pre> AEGP_ItemH itemH, AEGP_ItemFlags *item_flagsP); </pre> <p>Flag values (may be OR'd together):</p> <pre> 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 </pre>
AEGP_SetItemUseProxy	<p>Toggle item's proxy usage. Undoable.</p> <pre> *AEGP_SetItemUseProxy(AEGP_ItemH itemH, A_Boolean use_proxyB); </pre>
AEGP_GetItemParentFolder	<p>Get folder containing item.</p> <pre> AEGP_GetItemParentFolder(AEGP_ItemH itemH, AEGP_ItemH *parent_itemPH); </pre>
AEGP_SetItemParentFolder	<p>Sets an item's parent folder. Undoable.</p> <pre> AEGP_SetItemParentFolder(AEGP_ItemH itemH, AEGP_ItemH parent_folderH); </pre>
AEGP_GetItemDuration	<p>Get duration of item, in seconds.</p> <pre> AEGP_GetItemDuration(AEGP_ItemH itemH, A_Time *durationPT); </pre>
AEGP_GetItemCurrentTime	<p>Get current time within item. Not updated while rendering.</p> <pre> AEGP_GetItemCurrentTime(AEGP_ItemH itemH, A_long *curr_timePT); </pre>
AEGP_GetItemDimensions	<p>Get width and height of item.</p> <pre> AEGP_GetItemDimensions(AEGP_ItemH itemH, A_long *widthPL) A_long *heightPL); </pre>

TABLE 66: AEGP_ITEMSUITE8

Function	Purpose
<code>AEGP_GetItemPixelAspectRatio</code>	<p>Get the width of a pixel, assuming its height is 1.0, as numerator over denominator.</p> <pre> AEGP_GetItemPixelAspectRatio(AEGP_ItemH itemH, A_Ratio *ratioPRt); </pre>
<code>AEGP_DeleteItem</code>	<p>Removes item from all compositions. Undo-able. Do not use the <code>AEGP_ItemH</code> after calling this function.</p> <pre> AEGP_DeleteItem(AEGP_ItemH itemH); </pre>
<code>AEGP_GetItemSolidColor</code>	<p>Removed in AEGP_ItemSuite4. See AEGP_GetSolidFootageColor</p> <p>Given a solid item, return its color.</p> <pre> AEGP_GetItemSolidColor(AEGP_ItemH itemH, PF_Pixel *PF_Pixel); </pre>
<code>AEGP_SetSolidColor</code>	<p>Removed in AEGP_ItemSuite4. See AEGP_SetSolidFootageColor.</p> <p>Sets the color of an existing solid (error if itemH is not a solid).</p> <pre> AEGP_SetSolidColor(AEGP_ItemH itemH, AEGP_ColorVal color); </pre>
<code>AEGP_SetSolidDimensions</code>	<p>Removed in AEGP_ItemSuite4. See AEGP_SetSolidFootageDimensions.</p> <p>Sets the dimensions of an existing solid (error if itemH is not a solid).</p> <pre> AEGP_SetSolidDimensions(AEGP_ItemH itemH, A_short widthS, A_short heightS); </pre>
<code>AEGP_CreateNewFolder</code>	<p>Creates a new folder in the project. The newly created folder is allocated and owned by After Effects. Passing <code>NULL</code> for <code>parent_folderH0</code> creates the folder at the project's root.</p> <pre> AEGP_CreateNewFolder(const A_UTF16Char *nameZ, AEGP_ProjectH projH), AEGP_ItemH parentH0), AEGP_ItemH *new_folderPH); </pre>

TABLE 66: AEGP_ITEMSUITE8

Function	Purpose
AEGP_SetItemCurrentTime	Sets the current time within a given itemH. <pre>AEGP_SetItemCurrentTime(AEGP_ItemH itemH, const A_Time *new_timePT);</pre>
<u>AEGP_RenderNewItemSoundData()</u> <i>used to be here, but is now part of AEGP_RenderSuite.</i>	
AEGP_GetItemCommentLength	Retrieves the length (in characters) of the itemH's comment. <pre>AEGP_GetItemCommentLength(AEGP_ItemH itemH, A_u_long *buf_sizePLu);</pre>
AEGP_GetItemComment	Retrieves the itemH's comment. <pre>AEGP_GetItemComment(AEGP_ItemH itemH, A_u_long buf_sizeLu, A_char *commentZ);</pre>
AEGP_SetItemComment	Sets the itemH's comment. <pre>AEGP_SetItemComment(AEGP_ItemH itemH, const char *commentZ);</pre>
AEGP_GetItemLabel	Retrieves an item's label. <pre>AEGP_GetItemLabel(AEGP_ItemH itemH, AEGP_LabelID *labelP);</pre>
AEGP_SetItemLabel	Sets an item's label. <pre>AEGP_SetItemLabel(AEGP_ItemH itemH, AEGP_LabelID label);</pre>
AEGP_GetItemMRUView	New for CS3. 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. <pre>AEGP_SetItemLabel(AEGP_ItemH itemH, AEGP_ItemViewP *mru_viewP);</pre>

BLENDING TABLES

We've provided a function so AEGPs can obtain (but not examine) After Effects' current blending tables.

TABLE 67: AEGP_COLORSETTINGSUITE2

Function	Purpose
<code>AEGP_GetBlendingTables</code>	Retrieves the current <code>PF_EffectBlendingTables</code> , for use with AEGP_TransferRect . <code>AEGP_GetBlendingTables(PR_RenderContextH render_contextH, PF_EffectBlendingTables *blending_tables);</code>
<code>AEGP_DoesViewHaveColorSpaceXform</code>	Returns whether there is a colorspace transform applied to the current item view. <code>AEGP_DoesViewHaveColorSpaceXform(AEGP_ItemViewP viewP, A_Boolean *has_xformPB);</code>
<code>AEGP_XformWorkingToViewColorSpace</code>	Changes the view colorspace of the source to be the working colorspace of the destination. Source and destination can be the same. <code>AEGP_XformWorkingToViewColorSpace(AEGP_ItemViewP viewP, AEGP_WorldH srch, AEGP_WorldH dstH);</code>
<code>AEGP_GetNewWorkingSpaceColorProfile</code>	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. <code>AEGP_GetNewWorkingSpaceColorProfile(AEGP_PluginID aegp_plugin_id, AEGP_MemHandle *icc_profPH);</code>
<code>AEGP_GetNewColorProfileFromICCPProfile</code>	Retrieves a new <code>AEGP_ColorProfileP</code> from After Effects, representing the specified ICC profile. The caller must dispose of the returned <code>AEGP_ColorProfileP</code> using AEGP_DisposeColorProfile() . <code>AEGP_GetNewColorProfileFromICCPProfile(AEGP_PluginID aegp_plugin_id, A_long icc_sizeL, const void *icc_dataPV, AEGP_ColorProfileP *profilePP);</code>

TABLE 67: AEGP_COLORSETTINGS SUITE 2

Function	Purpose
<code>AEGP_GetNewICCPProfileFromColorProfile</code>	<p>Retrieves a new ICC profile (stored in an <code>AEGP_MemHandle</code>) representing the specified color profile. Returned <code>AEGP_MemHandle</code> must be disposed by the caller.</p> <pre> AEGP_GetNewICCPProfile FromColorProfile(AEGP_PluginID plugin_id, AEGP_ConstColorProfileP profileP, AEGP_MemHandle *profilePH); </pre>
<code>AEGP_GetNewColorProfileDescription</code>	<p>Returns a textual description of the specified color profile. Text will be a null-terminated UTF16 string, which must be disposed by the caller.</p> <pre> AEGP_GetNewColorProfileDescription(AEGP_PluginID aegp_plugin_id, AEGP_ConstColorProfileP profileP, AEGP_MemHandle *unicode_descPH); </pre>
<code>AEGP_DisposeColorProfile</code>	<p>Disposes of a color profile, obtained using other functions in this suite.</p> <pre> AEGP_DisposeColorProfile(AEGP_ColorProfileP profileP); </pre>
<code>AEGP_GetColorProfileApproximateGamma</code>	<p>Returns a floating point number approximating the gamma setting used by the specified color profile.</p> <pre> AEGP_GetColorProfileApproximateGama(AEGP_ConstColorProfileP profileP, A_FpShort *approx_gammaP); </pre>
<code>AEGP_IsRGBColorProfile</code>	<p>Returns whether the specified color profile is RGB.</p> <pre> AEGP_IsRGBColorProfile(AEGP_ConstColorProfileP profileP, A_Boolean *is_rgbPB); </pre>

MANIPULATE COMPOSITIONS

Provide information about the compositions in a project, and create cameras, lights, and solids.

TABLE 68: AEGP_COMPSUITE7

Function	Purpose
<code>AEGP_GetCompFromItem</code>	Retrieves the handle to the composition, given an item handle. Returns NULL if <code>itemH</code> is not an <code>AEGP_CompH</code> . <pre>AEGP_GetCompFromItem(AEGP_ItemH itemH, AEGP_CompH *compPH);</pre>
<code>AEGP_GetItemFromComp</code>	Used to get the item handle, given a composition handle. <pre>AEGP_GetItemFromComp(AEGP_CompH compH, AEGP_ItemH *itemPH);</pre>
<code>AEGP_GetCompDownsampleFactor</code>	Returns current downsample factor. Measured in pixels X by Y. Users can choose a custom downsample factor with independent X and Y. <pre>AEGP_GetCompDownsampleFactor(AEGP_CompH compH, AEGP_DownsampleFactor *dsfP);</pre>
<code>AEGP_SetCompDownsampleFactor</code>	Sets the composition's downsample factor. <pre>AEGP_SetCompDownsampleFactor(AEGP_CompH compH, AEGP_DownsampleFactor *dsfP);</pre>
<code>AEGP_GetCompBGColor</code>	Returns the composition background color. As of 7.0, the get/setters for composition background color take <code>AEGP_ColorVals</code> , not <code>PF_Pixels</code> . <pre>AEGP_GetCompBGColor(AEGP_CompH compH, AEGP_ColorVal *bg_colorP);</pre>
<code>AEGP_SetCompBGColor</code>	Sets a composition's background color. <pre>AEGP_SetCompBGColor(AEGP_CompH compH, const AEGP_ColorVal *bg_colorP);</pre>

TABLE 68: AEGP_COMPSUITE7

Function	Purpose
AEGP_GetCompFlags	<p>Returns composition flags, or'd together.</p> <pre>AEGP_GetCompFlags(AEGP_CompHcompH, AEGP_CompFlags *AEGP_CompFlags);</pre> <p>AEGP_CompFlag_SHOW_ALL_SHY AEGP_CompFlag_LOCK_SELECTION AEGP_CompFlag_LOCK_ALL_LAYERS AEGP_CompFlag_ENABLE_MOTION_BLUR AEGP_CompFlag_ENABLE_TIME_FILTER</p>
AEGP_GetCompFramerate	<p>Returns the composition's frames per second.</p> <pre>AEGP_GetCompFramerate(AEGP_CompH compH, A_FpLong *fpsPF);</pre>
AEGP_SetCompFramerate	<p>Sets the composition's frames per second.</p> <pre>AEGP_SetCompFramerate(AEGP_CompH compH, A_FpLong *fpsPF);</pre>
AEGP_GetCompShutterAnglePhase	<p>The composition shutter angle and phase.</p> <pre>AEGP_GetCompShutterAnglePhase(AEGP_CompH compH, A_Ratio *angle, A_Ratio *phase);</pre>
AEGP_GetCompShutterFrameRange	<p>The duration of the shutter frame, in seconds.</p> <pre>AEGP_GetCompShutterFrameRange(AEGP_CompH compH, const A_Time *comp_timeP);</pre>
AEGP_GetCompSuggestedMotionBlurSamples	<p>Retrieves the number of motion blur samples After Effects would perform in the given composition.</p> <pre>AEGP_GetCompSuggestedMotionBlurSamples(AEGP_CompH compH, A_long *samplesPL)</pre>
AEGP_SetCompSuggestedMotionBlurSamples	<p>Specified the number of motion blur samples After Effects will perform in the given composition. Undoable.</p> <pre>AEGP_SetCompSuggestedMotionBlurSamples(AEGP_CompH compH, A_long samplesL);</pre>

TABLE 68: AEGP_COMPSUITE7

Function	Purpose
<code>AEGP_GetCompWorkAreaStart</code>	<p>Get the time where the current work area starts.</p> <pre>AEGP_GetCompWorkAreaStart(AEGP_CompH compH, A_Time *startPT);</pre>
<code>AEGP_GetCompWorkAreaDuration</code>	<p>Get the duration of a composition's current work area, in seconds.</p> <pre>AEGP_GetCompWorkAreaDuration(AEGP_CompH compH, A_Time *durationPT);</pre>
<code>AEGP_SetCompWorkAreaStartAndDuration</code>	<p>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.</p> <pre>AEGP_SetCompWorkAreaStartAndDuration(AEGP_CompH compH, const A_Time *startPT) const A_Time *durationPT);</pre>
<code>AEGP_CreateSolidInComp</code>	<p>Creates a new solid with a specified width, height, color, and duration in the composition. Undo-able.</p> <p>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.</p> <pre>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);</pre>
<code>AEGP_CreateCameraInComp</code>	<p>Creates and adds a camera to the specified composition. Once created, you can manipulate the camera's parameter streams using the <code>AEGP_StreamSuite</code>.</p> <pre>AEGP_CreateCameraInComp(const A_UTF16Char *utf_nameZ, A_FloatPoint center_point, AEGP_CompH parent_compH, AEGP_LayerH *new_cameraPH);</pre>

TABLE 68: AEGP_COMPSUITE7

Function	Purpose
AEGP_CreateLightInComp	<p>Creates and adds a light to the specified composition. Once created, you can manipulate the light's parameter streams using the AEGP_StreamSuite.</p> <pre>AEGP_CreateLightInComp(const A_UTF16Char *utf_nameZ, A_FloatPoint center_point, AEGP_CompH parent_compH, AEGP_LayerH *new_lightPH);</pre>
AEGP_CreateComp	<p>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.</p> <pre>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);</pre>
AEGP_GetNewCollectionFromCompSelection	<p>Creates a new AEGP_Collection2H from the items selected in the given composition. The plug-in is responsible for disposing of the AEGP_Collection2H.</p> <pre>AEGP_GetNewCollectionFromCompSelection(AEGP_PluginID plugin_id, AEGP_CompH compH, AEGP_Collection2H *collectionPH);</pre>
AEGP_SetSelection	<p>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.</p> <pre>AEGP_SetSelection(AEGP_CompH compH, AEGP_Collection2H collectionH);</pre>

TABLE 68: AEGP_COMPSUITE7

Function	Purpose
<code>AEGP_SetCompDisplayStartTime</code>	<p>Not undo-able. Sets the displayed start time of a composition (has no effect on the duration of the composition).</p> <pre>AEGP_SetCompDisplayStartTime(AEGP_CompH compH, const A_Time *start_timePT);</pre>
<code>AEGP_SetCompDuration</code>	<p>Undoable. Sets the duration of the given composition.</p> <pre>AEGP_SetCompDuration(AEGP_CompH compH, const A_Time *durationPT);</pre>
<code>AEGP_CreateNullInComp</code>	<p>Creates a “null object” in the composition (useful for translating projects from 3D applications into After Effects).</p> <p>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.</p> <pre>AEGP_CreateNullInComp(const A_UTF16Char *utf_nameZ, AEGP_CompH parent_compH, const A_Time *durationPT0, AEGP_LayerH *new_null_solidPH);</pre>
<code>AEGP_SetCompPixelAspectRatio</code>	<p>Sets the pixel aspect ratio of a composition.</p> <pre>AEGP_SetCompPixelAspectRatio(AEGP_CompH compH, const A_Ratio *parPRt);</pre>
<code>AEGP_CreateTextLayerInComp</code>	<p>Creates a text layer in the composition, and returns its <code>AEGP_LayerH</code>.</p> <pre>AEGP_CreateTextLayerInComp(AEGP_CompH parent_compH, AEGP_LayerH *new_text_lyrPH);</pre>
<code>AEGP_SetCompDimensions</code>	<p>Sets the dimensions of the composition. Undoable.</p> <pre>AEGP_SetCompDimensions(AEGP_CompH compH, A_long widthL, A_long heightL);</pre>
<code>AEGP_DuplicateComp</code>	<p>Duplicates the composition. Undoable.</p> <pre>AEGP_DuplicateComp(AEGP_CompH compH, AEGP_CompH *new_compPH);</pre>

TABLE 68: AEGP_COMPSUITE7

Function	Purpose
AEGP_GetCompFrameDuration	Retrieves the duration of a frame in a composition. <pre>AEGP_GetCompFrameDuration(AEGP_CompH compH, A_Time *timeP);</pre>
AEGP_GetMostRecentlyUsedComp	New in 7.0. Returns the most-recently-used composition. <pre>AEGP_GetMostRecentlyUsedComp(AEGP_CompH *compPH);</pre>
AEGP_CreateVectorLayerInComp	Creates and returns a handle to a new vector layer. <pre>AEGP_CreateVectorLayerInComp(AEGP_CompH parent_compH, AEGP_LayerH *new_vec_layerPH);</pre>
AEGP_GetNewCompMarkerStream	Returns an AEGP_StreamRefH to the composition's marker stream. Must be disposed by caller. <pre>AEGP_GetNewCompMarkerStream(AEGP_PluginID aegp_plugin_id, AEGP_CompH parent_compH, AEGP_StreamRefH *streamPH);</pre>

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 69: 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. <pre>AEGP_GetMainFootageFromItem(AEGP_ItemH itemH, AEGP_FootageH *footagePH);</pre>

TABLE 69: AEGP_FOOTAGESUITE5

Function	Purpose
<code>AEGP_GetProxyFootageFromItem</code>	<p>Returns an error if item has no proxy. Returns the proxy footage handle. Note: a composition can have a proxy.</p> <pre> AEGP_GetProxyFootageFromItem(AEGP_ItemH itemH, AEGP_FootageH *proxy_ftgPH); </pre>
<code>AEGP_GetFootageNumFiles</code>	<p>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).</p> <pre> AEGP_GetFootageNumFiles(AEGP_FootageH footageH, A_long *num_filesPL0, A_long *files_per_frmPL0); </pre>
<code>AEGP_GetFootagePath</code>	<p>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). <code>frame_numL</code> ranges from 0 to <code>num_main_files</code>, as obtained using AEGP_GetFootageNumFiles. <code>AEGP_FOOTAGE_MAIN_FILE_INDEX</code> is the main file. The path is a handle to a NULL-terminated <code>A_UTF16Char</code> string, and must be disposed with <code>AEGP_FreeMemHandle</code>.</p> <pre> AEGP_GetFootagePath(AEGP_FootageH footageH, A_long frame_numL, A_long file_indexL, AEGP_MemHandle *unicode_pathPH); </pre>
<code>AEGP_GetFootageSignature</code>	<p>Retrieves the footage signature of specified footage.</p> <pre> AEGP_GetFootageSignature(AEGP_FootageH footageH, AEGP_FootageSignature *sigP); </pre> <p>The signature will be one of the following:</p> <pre> AEGP_FootageSignature_NONE AEGP_FootageSignature_MISSING AEGP_FootageSignature_SOLID </pre>

TABLE 69: AEGP_FOOTAGESUITE5

Function	Purpose
AEGP_NewFootage	<p>Creates a new footage item. The file path is a NULL-terminated UTF-16 string with platform separators.</p> <pre> 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); </pre> <p>Note the optional params. If allow_interpretation_dialogB is FALSE, After Effects will guess the alpha interpretation.</p> <pre> typedef struct { A_long layer_idL; A_long layer_indexL; char *nameAC; AEGP_LayerDrawStyle draw_style; } AEGP_FootageLayerKey; </pre> <p>AEGP_LayerDrawStyle can be:</p> <pre> AEGP_LayerDrawStyle_LAYER_BOUNDS AEGP_LayerDrawStyle_DOCUMENT_BOUNDS </pre> <p>AEGP_InterpretationStyle can be:</p> <pre> AEGP_InterpretationStyle_NO_DIALOG_GUESS </pre> <p>Will guess alpha interpretation even if file contains unknown alpha interpretation and user pref says to ask user. Equivalent to FALSE for backwards compatibility with CS4 and earlier.</p> <pre> AEGP_InterpretationStyle_DIALOG_OK </pre> <p>Optionally can show a dialog. Equivalent to TRUE for backwards compatibility with CS4 and earlier.</p> <pre> AEGP_InterpretationStyle_NO_DIALOG_NO_GUESS </pre> <p>New in CS5. Used for replace footage implementation.</p>

TABLE 69: AEGP_FOOTAGESUITE5

Function	Purpose
<code>AEGP_AddFootageToProject</code>	<p>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.</p> <pre> AEGP_AddFootageToProject(AEGP_FootageH footageH, AEGP_ItemH folderH, AEGP_ItemH *add_itemPH0); </pre>
<code>AEGP_SetItemProxyFootage</code>	<p>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.</p> <pre> AEGP_SetItemProxyFootage(AEGP_FootageH footageH, AEGP_ItemH itemH); </pre>
<code>AEGP_ReplaceItemMainFootage</code>	<p>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.</p> <pre> AEGP_ReplaceItemMainFootage(AEGP_FootageH footageH, AEGP_ItemH itemH); </pre>
<code>AEGP_DisposeFootage</code>	<p>Deletes a footage item. Do not dispose of footage you did not create, or that has been added to the project.</p> <pre> AEGP_DisposeFootage(AEGP_FootageH footageH); </pre>
<code>AEGP_GetFootageInterpretation</code>	<p>Populates an <code>AEGP_FootageInterp</code> describing the settings of the <code>AEGP_FootageH</code>. There is no way to create a valid <code>AEGP_FootageInterp</code> other than by using this function.</p> <pre> AEGP_GetFootageInterpretation(const AEGP_ItemH itemH, A_Boolean proxyB, AEGP_FootageInterp *interpP); </pre> <p>If <code>proxyB</code> is <code>TRUE</code>, the proxy footage's settings are retrieved.</p>

TABLE 69: AEGP_FOOTAGESUITE5

Function	Purpose
<code>AEGP_SetFootageInterpretation</code>	<p>Apply the settings in the <code>AEGP_FootageInterp</code> to the <code>AEGP_FootageH</code>. Undo-able.</p> <pre>AEGP_SetFootageInterpretation(const AEGP_ItemH itemH, A_Boolean proxyB, const AEGP_FootageInterp *interpP);</pre> <p>If <code>proxyB</code> is <code>TRUE</code>, the proxy footage's settings are modified.</p>
<code>AEGP_GetFootageLayerKey</code>	<p>Populates an <code>AEGP_FootageLayerKey</code> describing the footage.</p> <pre>AEGP_GetFootageLayerKey(AEGP_FootageH footageH, AEGP_FootageLayerKey*layerKeyP);</pre>
<code>AEGP_NewPlaceholderFootage</code>	<p>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.</p> <pre>AEGP_NewPlaceholderFootage(AEGP_PluginID plugin_id, const A_char *nameZ, A_long width, A_long height, const A_Time *durationPT, AEGP_FootageH *footagePH);</pre>
<code>AEGP_NewPlaceholderFootageWithPath</code>	<p>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.</p> <pre>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);</pre>

TABLE 69: AEGP_FOOTAGESUITE5

Function	Purpose
<code>AEGP_NewSolidFootage</code>	<p>As of 6.0, this is the way to add a solid. Until the footage is added to the project, the caller owns the <code>AEGP_FootageH</code> (and must dispose of it if, and only if, it isn't added to the project).</p> <pre> AEGP_NewSolidFootage(const A_char *nameZ, A_long width, A_long height, const AEGP_ColorVal *colorP, AEGP_FootageH *footagePH); </pre>
<code>AEGP_GetSolidFootageColor</code>	<p>Returns the color of a given solid. Returns an error if the <code>AEGP_ItemH</code> is not a solid.</p> <pre> AEGP_GetSolidFootageColor(AEGP_ItemH itemH, A_Boolean proxyB, AEGP_ColorVal *colorP); </pre> <p>If <code>proxyB</code> is TRUE, the proxy solid's color is retrieved.</p>
<code>AEGP_SetSolidFootageColor</code>	<p>Sets the color of a solid. Undo-able.</p> <pre> AEGP_SetSolidFootageColor(AEGP_ItemH itemH, A_Boolean proxyB, AEGP_ColorVal *colorP); </pre> <p>If <code>proxyB</code> is TRUE, the proxy solid's color is set.</p>
<code>AEGP_SetSolidFootageDimensions</code>	<p>Sets the dimensions of a solid. Undo-able.</p> <pre> AEGP_SetSolidFootageDimensions(AEGP_ItemH itemH, A_Boolean proxyB, A_long widthL, A_long heightL); </pre> <p>If <code>proxyB</code> is TRUE, the proxy solid's dimensions are modified. Returns an error if the item isn't a solid.</p>
<code>AEGP_GetFootageSoundDataFormat</code>	<p>Retrieves information about the audio data in the footage item (by populating the <code>AEGP_SoundDataFormat</code> you passed in).</p> <pre> AEGP_GetFootageSoundDataFormat(AEGP_FootageH footageH, AEGP_SoundDataFormat *formatP); </pre>

TABLE 69: AEGP_FOOTAGESUITE5

Function	Purpose
<code>AEGP_GetFootageSequenceImportOptions</code>	<p>Populates and returns a <code>AEGP_FileSequenceImportOptions</code> describing the given <code>AEGP_FootageH</code>.</p> <pre>AEGP_GetFootageSequenceImportOptions(AEGP_FootageH footageH, AEGP_FileSequenceImportOptions *optionsP);</pre>

TABLE 70: AEGP_FOOTAGEINTERP STRUCTURE

Member	Purpose
<code>AEGP_InterlaceLabel il;</code>	<p>The interlace settings for the footage item.</p> <pre>A_u_long signature; // 'FIEL' A_short version; FIEL_Type type; FIEL_Order order; A_u_long reserved;</pre> <p><code>FIEL_Type</code> is one of the following:</p> <pre>FIEL_Type_FRAME_RENDERED FIEL_Type_INTERLACED FIEL_Type_HALF_HEIGHT FIEL_Type_FIELD_DOUBLED</pre> <p><code>FIEL_Type_FIELD_DOUBLED</code> means 60 full-sized field doubled frames per second.</p> <p><code>FIEL_Order</code> is either <code>FIEL_Order_UPPER_FIRST</code> or <code>FIEL_Order_LOWER_FIRST</code>.</p>
<code>AEGP_AlphaLabel al;</code>	<pre>AEGP_AlphaFlag flags; A_u_char redCu; A_u_char greenCu; A_u_char blueCu;</pre> <p><code>AEGP_AlphaFlags</code> is one or more of the following, OR'd together:</p> <pre>AEGP_AlphaPremul AEGP_AlphaInverted AEGP_AlphaIgnore</pre> <p>If <code>AEGP_AlphaPremul</code> is not set, straight alpha is assumed. <code>AEGP_AlphaInverted</code> indicates that higher values are transparent, instead of lower.</p>

TABLE 70: AEGP_FOOTAGEINTERP STRUCTURE

Member	Purpose
<code>AEGP_PulldownPhase pd;</code>	Indicates the phase for use in 3:2 pulldown. One of the following: <code>AEGP_PulldownPhase_NO_PULLDOWN,</code> <code>AEGP_PulldownPhase_WSSWW,</code> <code>AEGP_PulldownPhase_SSWWW,</code> <code>AEGP_PulldownPhase_SWWWS,</code> <code>AEGP_PulldownPhase_WWWSS,</code> <code>AEGP_PulldownPhase_WWSSW,</code> <code>AEGP_PulldownPhase_WWWSW,</code> <code>AEGP_PulldownPhase_WWSWW,</code> <code>AEGP_PulldownPhase_WSWWW,</code> <code>AEGP_PulldownPhase_SWWWW,</code> <code>AEGP_PulldownPhase_WWWWS</code>
<code>AEGP_LoopBehavior loop;</code>	Indicates the number of times the footage should loop. <code>A_long loops;</code> <code>A_long reserved;</code>
<code>A_Ratio pix_aspect_ratio;</code>	Expresses the pixel aspect ratio of the footage (x over y).
<code>A_FpLong native_fpsF;</code>	The original framerate (in frames per second) of the footage item.
<code>A_FpLong conform_fpsF;</code>	The framerate being used for the footage item.
<code>A_long depthL;</code>	The pixel depth of the footage. One of the following: <code>AEGP_Footage_Depth_1</code> <code>AEGP_Footage_Depth_2</code> <code>AEGP_Footage_Depth_4</code> <code>AEGP_Footage_Depth_8</code> <code>AEGP_Footage_Depth_16</code> <code>AEGP_Footage_Depth_24</code> <code>AEGP_Footage_Depth_30</code> <code>AEGP_Footage_Depth_32</code> <code>AEGP_Footage_Depth_GRAY_2</code> <code>AEGP_Footage_Depth_GRAY_4</code> <code>AEGP_Footage_Depth_GRAY_8</code> <code>AEGP_Footage_Depth_48</code> <code>AEGP_Footage_Depth_64</code> <code>AEGP_Footage_Depth_GRAY_16</code>
<code>A_Boolean motion_dB;</code>	Indicates whether motion de-interlacing is being applied to the footage item.

WORK WITH AUDIO

`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 71: AEGP_SOUNDDATASUITE1

Function	Purpose
<code>AEGP_NewSoundData</code>	Creates a new <code>AEGP_SoundDataH</code> , of which the plug-in must dispose. <code>AEGP_NewSoundData(const AEGP_SoundDataFormat *formatP, AEGP_SoundDataH *new_dataH);</code>
<code>AEGP_DisposeSoundData</code>	Frees an <code>AEGP_SoundDataH</code> . <code>AEGP_DisposeSoundData(AEGP_SoundDataH sound_dataH);</code>
<code>AEGP_GetSoundDataFormat</code>	Obtains information about the format of a given <code>AEGP_SoundDataH</code> . <code>AEGP_GetSoundDataFormat(AEGP_SoundDataH soundH, AEGP_SoundDataFormat *formatP);</code>
<code>AEGP_LockSoundDataSamples</code>	Locks the <code>AEGP_SoundDataH</code> in memory. <code>AEGP_LockSoundDataSamples(AEGP_SoundDataH soundH, void **samples);</code>
<code>AEGP_UnlockSoundDataSamples</code>	Unlocks an <code>AEGP_SoundDataH</code> . <code>AEGP_UnlockSoundDataSamples(AEGP_SoundDataH soundH);</code>
<code>AEGP_GetNumSamples</code>	Obtains the number of samples in the given <code>AEGP_SoundDataH</code> . <code>AEGP_GetNumSamples(AEGP_SoundDataH soundH, A_long *numsamplesPL);</code>

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 72: AEGP_LAYERSUITE7

Function	Purpose
AEGP_GetCompNumLayers	Obtains the number of layers in a composition. <code>AEGP_GetCompNumLayers(AEGP_CompH compH, A_long *num_layersPL);</code>
AEGP_GetCompLayerByIndex	Get a AEGP_LayerH from a composition. Zero is the foremost layer. <code>AEGP_GetCompLayerByIndex(AEGP_CompH compH, A_long layer_indexL, AEGP_LayerH *layerPH);</code>
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). <code>AEGP_GetActiveLayer(AEGP_LayerH *layerPH);</code>
AEGP_GetLayerIndex	Get the index of the layer (0 is the topmost layer in the composition). <code>AEGP_GetLayerIndex(AEGP_LayerH layerH, A_long *layer_indexPL);</code>
AEGP_GetLayerSourceItem	Get the AEGP_ItemH of the layer's source item. <code>AEGP_GetLayerSourceItem(AEGP_LayerH layerH, AEGP_ItemH *source_itemPH);</code>
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. <code>AEGP_GetLayerSourceItemID(AEGP_LayerH layerH, A_long *source_idPL);</code>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
<code>AEGP_GetLayerParentComp</code>	<p>Get the <code>AEGP_CompH</code> of the composition containing the layer.</p> <pre> AEGP_GetLayerParentComp(AEGP_LayerH layerH, AEGP_CompH *compPH); </pre>
<code>AEGP_GetLayerName</code>	<p>Get the name of a layer. Both <code>utf_layer_namePH0</code> and <code>utf_source_namePH0</code> point to null terminated UTF-16 strings. They must be disposed with <code>AEGP_FreeMemHandle</code>.</p> <pre> AEGP_GetLayerName(AEGP_PluginID pluginID, AEGP_LayerH layerH, AEGP_MemHandle *utf_layer_namePH0, AEGP_MemHandle *utf_source_namePH0); </pre>
<code>AEGP_GetLayerQuality</code>	<p>Get the quality of a layer. Layer quality is one of the following flags:</p> <pre> AEGP_GetLayerQuality(AEGP_LayerH layerH, AEGP_LayerQuality *qualityP); </pre> <p> <code>AEGP_LayerQual_NONE</code> <code>AEGP_LayerQual_WIREFRAME</code> <code>AEGP_LayerQual_DRAFT</code> <code>AEGP_LayerQual_BEST</code> </p>
<code>AEGP_SetLayerQuality</code>	<p>Sets the quality of a layer (see flag values above). Undoable.</p> <pre> AEGP_SetLayerQuality(AEGP_LayerH layerH, AEGP_LayerQuality quality); </pre>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
AEGP_GetLayerFlags	<p>Get flags for a layer.</p> <pre> 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 </pre>
AEGP_SetLayerFlag	<p>Sets one layer flag at a time. Undoable.</p> <pre> AEGP_SetLayerFlag(AEGP_LayerH layerH, AEGP_LayerFlags single_flag, A_Boolean valueB); </pre>
AEGP_IsLayerVideoReallyOn	<p>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.</p> <pre> AEGP_IsLayerVideoReallyOn(AEGP_LayerH layerH, A_Boolean *onPB); </pre>
AEGP_IsLayerAudioReallyOn	<p>Accounts for solo status of other layers in the composition.</p> <pre> AEGP_IsLayerAudioReallyOn(AEGP_LayerH layerH, A_Boolean *onPB); </pre>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
<code>AEGP_GetLayerCurrentTime</code>	<p>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.</p> <pre> AEGP_GetLayerCurrentTime(AEGP_LayerH layerH, AEGP_LTimeMode time_mode, A_Time *curr_timePT); </pre>
<code>AEGP_GetLayerInPoint</code>	<p>Get time of first visible frame in composition or layer time. In layer time, the <code>in_pointPT</code> is always 0.</p> <pre> AEGP_GetLayerInPoint(AEGP_LayerH layerH, AEGP_LTimeMode time_mode, A_Time *in_pointPT); </pre>
<code>AEGP_GetLayerDuration</code>	<p>Get duration of layer, in composition or layer time, in seconds.</p> <pre> AEGP_GetLayerDuration(AEGP_LayerH layerH, AEGP_LTimeMode time_mode, A_Time *durationPT); </pre>
<code>AEGP_SetLayerInPointAndDuration</code>	<p>Set duration and in point of layer in composition or layer time. Undo-able.</p> <pre> AEGP_SetLayerInPointAndDuration(AEGP_LayerH layerH, AEGP_LTimeMode time_mode, const A_Time *in_pointPT, const A_Time *durationPT); </pre>
<code>AEGP_GetLayerOffset</code>	<p>Get the offset from the start of the composition to layer time 0, in composition time.</p> <pre> AEGP_GetLayerOffset(AEGP_LayerH layerH, A_Time *offsetPT); </pre>
<code>AEGP_SetLayerOffset</code>	<p>Set the offset from the start of the composition to the first frame of the layer, in composition time. Undoable.</p> <pre> AEGP_SetLayerOffset(AEGP_LayerH layerH, A_Time *offsetPT); </pre>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
AEGP_GetLayerStretch	Get stretch factor of a layer. <pre>AEGP_GetLayerStretch(AEGP_LayerH layerH, A_Ratio *stretchPrt);</pre>
AEGP_SetLayerStretch	Set stretch factor of a layer. <pre>AEGP_SetLayerStretch(AEGP_LayerH layerH, A_Ratio *stretchPrt);</pre>
AEGP_GetLayerTransferMode	Get transfer mode of a layer. <pre>AEGP_GetLayerTransferMode(AEGP_LayerH layerH, AEGP_LayerTransferMode *modeP);</pre>
AEGP_SetLayerTransferMode	Set transfer mode of a layer. Undoable. <pre>AEGPSetLayerTransferMode(AEGP_LayerH layerH, AEGP_LayerTransferMode *modeP);</pre> <p>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.</p>
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. <pre>AEGP_IsAddLayerValid(AEGP_ItemH item_to_addH, AEGP_CompH into_compH, A_Boolean *validPB);</pre>
AEGP_AddLayer	Add an item to the composition, above all other layers. Undo-able. Use AEGP_IsAddLayerValid() first, to confirm that it's possible. <pre>AEGP_AddLayer(AEGP_ItemH item_to_addH, AEGP_CompH into_compH, A_Boolean *added_layerPH0);</pre>
AEGP_ReorderLayer	Change the order of layers. Undoable. <pre>AEGP_ReorderLayer(AEGP_LayerH layerH, A_long layer_indexL);</pre>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
<code>AEGP_GetLayerMaskedBounds</code>	<p>Given a layer's handle and a time, returns the bounds of area visible with masks applied.</p> <pre> AEGP_GetLayerMaskedBounds(AEGP_LayerH layerH, const A_Time *comp_timePT, A_FloatRect *boundsPR); </pre>
<code>AEGP_GetLayerObjectType</code>	<p>Returns a layer's object type.</p> <pre> AEGP_GetLayerObjectType(AEGP_LayerH layerH, AEGP_ObjectType *object_type); </pre> <p> <code>AEGP_ObjectType_AV</code> <code>AEGP_ObjectType_LIGHT</code> <code>AEGP_ObjectType_CAMERA</code>, <code>AEGP_ObjectType_TEXT</code> </p>
<code>AEGP_IsLayer3D</code>	<p>Is the footage item a 3D layer. All AV layers are either 2D or 3D.</p> <pre> AEGP_IsLayer3D(AEGP_LayerH layerH, A_Boolean *is_3DPB); </pre>
<code>AEGP_IsLayer2D</code>	<p>Is the footage item a 2D layer. All AV layers are either 2D or 3D.</p> <pre> AEGP_IsLayer2D(AEGP_LayerH layerH, A_Boolean *is_2DPB); </pre>
<code>AEGP_IsVideoActive</code>	<p>Given composition time and a layer, see if the layer will render. Time mode is either <code>AEGP_LTimeMode_LayerTime</code> or <code>AEGP_LTimeMode_CompTime</code>.</p> <pre> AEGP_IsVideoActive(AEGP_LayerH layerH, AEGP_LTimeMode time_mode, A_Time *comp_timePT, A_Boolean *is_activePB); </pre>
<code>AEGP_IsLayerUsedAsTrackMatte</code>	<p>Is the layer used as a track matte?</p> <pre> AEGP_IsLayerUsedAsTrackMatte(AEGP_LayerH layerH, A_Boolean fill_must_be_activeB, A_Boolean *is_track_mattePB); </pre>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
<code>AEGP_DoesLayerHaveTrackMatte</code>	Does this layer have a Track Matte? <pre>AEGP_DoesLayerHaveTrackMatte(AEGP_LayerH layerH, A_Boolean *has_track_mattePB);</pre>
<code>AEGP_ConvertCompToLayerTime</code>	Given a time in composition space, returns the time relative to the layer source footage. <pre>AEGP_ConvertCompToLayerTime(AEGP_LayerH layerH, const A_Time *comp_timeP, A_Time *layer_timeP);</pre>
<code>AEGP_ConvertLayerToCompTime</code>	Given a time in layer space, find the corresponding time in composition space. <pre>AEGP_ConvertLayerToCompTime(AEGP_LayerH layerH, const A_Time *layer_timePT, A_Time *comp_timePT);</pre>
<code>AEGP_GetLayerDancingRandValue</code>	Used by the dancing dissolve transfer function. <pre>AEGP_GetLayerDancingRandValue(AEGP_LayerH layerH, const A_Time *comp_timePT, A_long *rand_valuePL);</pre>
<code>AEGP_GetLayerID</code>	Supplies the layer's unique ID. This ID never changes during the lifetime of the project. <pre>AEGP_GetLayerID(AEGP_LayerH layerH, AEGP_LayerIDVal *id_valP);</pre>
<code>AEGP_GetLayerToWorldXform</code>	Given a layer handle and time, returns the layer to world transformation matrix. <pre>AEGP_GetLayerToWorldXform(AEGP_LayerH aegp_layerH, const A_Time *comp_timeP, A_Matrix4 *transform);</pre>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
<code>AEGP_GetLayerToWorldXformFromView</code>	<p>Given a layer and composition handle and the current (composition) time, returns the translation between the user's view and the layer, corrected for the composition's current aspect ratio.</p> <pre> AEGP_GetLayerToWorldXformFromView(AEGP_LayerH aegp_layerH, const A_Time *view_timeP, const A_Time *comp_timeP, A_Matrix4 *transform); </pre>
<code>AEGP_SetLayerName</code>	<p>Sets the name of a layer. Undo-able. <code>new_nameZ</code> points to a null terminated UTF-16 string.</p> <pre> AEGP_SetLayerName(AEGP_LayerH aegp_layerH, const A_UTF16Char *new_nameZ); </pre>
<code>AEGP_GetLayerParent</code>	<p>Retrieves the handle to a layer's parent (none if not parented).</p> <pre> AEGP_GetLayerParent(AEGP_LayerH layerH, AEGP_LayerH *parent_layerPH); </pre>
<code>AEGP_SetLayerParent</code>	<p>Sets a layer's parent layer.</p> <pre> AEGP_SetLayerParent(AEGP_LayerH layerH, const AEGP_LayerH parent_layerH); </pre>
<code>AEGP_DeleteLayer</code>	<p>Deletes a layer. Can you believe it took us three suite versions to add a delete function? Neither can we.</p> <pre> AEGP_DeleteLayer(AEGP_LayerH layerH); </pre>
<code>AEGP_DuplicateLayer</code>	<p>Duplicates the layer. Undoable.</p> <pre> AEGP_DuplicateLayer(AEGP_LayerH orig_layerH, AEGP_LayerH *dupe_layerPH); </pre>
<code>AEGP_GetLayerFromLayerID</code>	<p>New in 7.0. Retrieves the <code>AEGP_LayerH</code> associated with a given <code>AEGP_LayerIDVal</code> (which is what you get when accessing an effect's layer parameter stream).</p> <pre> AEGP_GetLayerFromLayerID(AEGP_CompH parent_compH, AEGP_LayerIDVal id, AEGP_LayerH *layerPH); </pre>

TABLE 72: AEGP_LAYERSUITE7

Function	Purpose
<code>AEGP_GetLayerLabel</code>	New in CS5. Gets a layer's <code>AEGP_LabelID</code> . <pre>AEGP_GetLayerLabel(AEGP_LayerH layerH, AEGP_LabelID *labelP);</pre>
<code>AEGP_SetLayerLabel</code>	New in CS5. Sets a layer's <code>AEGP_LabelID</code> . Undoable. <pre>AEGP_SetLayerLabel(AEGP_LayerH layerH, AEGP_LabelID label);</pre>

A NOTE ABOUT LAYER OFFSETS

When the layer stretch factor (obtained using [AEGP_GetLayerStretch](#), naturally) is not 100%, the following computation will be needed to yield the correct layer offset:

```
offset = compIn - stretch * layerIn;
```

WORKING WITH STREAMS

Just about everything in After Effects is a stream. Masks, effect parameters, layers, geometrics, text and paint stroke layers are all internally represented by streams. The AEGP API can access nearly every aspect of every stream.

STREAM SUITE

Access and manipulate the values of a layer's streams.

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
<code>AEGP_IsStreamLegal</code>	Determines if the given stream is appropriate for the given layer. <pre>AEGP_IsStreamLegal(AEGP_LayerH layerH, AEGP_LayerStream which_stream, A_Boolean* is_legalP);</pre>

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
AEGP_CanVaryOverTime	<p>Given a stream, returns whether or not a stream is time-variant (and can be keyframed).</p> <pre> AEGP_CanVaryOverTime(AEGP_StreamRefH streamH, A_Boolean *can_varyPB); </pre>
AEGP_GetValidInterpolations	<p>Retrieves an AEGP_KeyInterpolationMask indicating which interpolation types are valid for the AEGP_StreamRefH.</p> <pre> AEGP_GetValidInterpolations(AEGP_StreamRefH streamH, AEGP_KeyInterpolationMask *valid_interp); </pre> <p>AEGP_KeyInterpolationMask will be a combination of the following:</p> <pre> AEGP_KeyInterpMask_NONE AEGP_KeyInterpMask_LINEAR AEGP_KeyInterpMask_BEZIER AEGP_KeyInterpMask_HOLD AEGP_KeyInterpMask_CUSTOM AEGP_KeyInterpMask_ANY </pre>

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
AEGP_GetNewLayerStream	<p>Get a layer's data stream. Plug-in must dispose of streamPH. Note that this will not provide keyframe access; Use the AEGP_KeyframeSuite instead.</p> <p> AEGP_LayerStream_ANCHORPOINT AEGP_LayerStream_POSITION AEGP_LayerStream_SCALE AEGP_LayerStream_ROTATION AEGP_LayerStream_OPACITY AEGP_LayerStream_AUDIO AEGP_LayerStream_MARKER AEGP_LayerStream_TIME_REMAP AEGP_LayerStream_ROTATE_X, AEGP_LayerStream_ROTATE_Y, AEGP_LayerStream_FOCAL_LENGTH, AEGP_LayerStream_DEPTH_OF_FIELD, AEGP_LayerStream_FOCAL_DISTANCE, AEGP_LayerStream_APERTURE, AEGP_LayerStream_BLUR_LEVEL, AEGP_LayerStream_INTENSITY, AEGP_LayerStream_COLOR, AEGP_LayerStream_CONE_ANGLE, AEGP_LayerStream_CONE_FEATHER, AEGP_LayerStream_SHADOW_DARKNESS, AEGP_LayerStream_SHADOW_DIFFUSION, AEGP_LayerStream_ACCEPTS_SHADOWS, AEGP_LayerStream_ACCEPTS_LIGHTS, AEGP_LayerStream_AMBIENT_COEFF, AEGP_LayerStream_DIFFUSE_COEFF, AEGP_LayerStream_SPECULAR_COEFF, AEGP_LayerStream_SHININESS_COEFF, AEGP_LayerStream_CASTS_SHADOWS, AEGP_LayerStream_LIGHT_TRANSMISSION, AEGP_LayerStream_METAL_COEFF, AEGP_LayerStream_SOURCE_TEXT </p> <pre> AEGP_GetNewLayerStream(AEGP_PluginID id, AEGP_LayerH layerH, AEGP_LayerStream which_stream, AEGP_StreamRefH *streamPH); </pre>
AEGP_GetEffectNumParamStreams	<p>Get number of parameter streams associated with an effect.</p> <pre> AEGP_GetEffectNumParamStreams(AEGP_EffectRefH effect_refH, A_long *num_parmsPL); </pre>

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
AEGP_GetNewEffectStreamByIndex	<p>Get an effect's parameter stream. Plug-in must dispose of streamPH.</p> <pre> AEGP_GetNewEffectStreamByIndex(AEGP_PluginID id, AEGP_EffectRefH effect_refH, PF_ParamIndex param_index, AEGP_StreamRefH *streamPH); </pre>
AEGP_GetNewMaskStream	<p>Get a mask's stream. The stream must be disposed. Also see the AEGP_MaskSuite and AEGP_MaskOutlineSuite for additional Mask functions.</p> <p>AEGP_MaskStream_OUTLINE, AEGP_MaskStream_OPACITY, AEGP_MaskStream_FEATHER, AEGP_MaskStream_EXPANSION,</p> <p>Useful for iteration:</p> <pre> AEGP_MaskStream_BEGIN = AEGP_MaskStream_OUTLINE, AEGP_MaskStream_END = AEGP_MaskStream_EXPANSION+1 </pre> <pre> AEGP_GetNewMaskStream(AEGP_PluginID aegp_plugin_id, AEGP_MaskRefH mask_refH, AEGP_MaskStream which_stream, AEGP_StreamRefH *mask_strmPH); </pre>
AEGP_DisposeStream	<p>Dispose of a stream (do this with all streams passed to the plug-in by these functions).</p> <pre> AEGP_DisposeStream(AEGP_StreamRefH streamH); </pre>
AEGP_GetNewMaskOpacity	<p>Get the mask's opacity stream. The stream must be disposed.</p> <pre> AEGP_GetNewMaskOpacity(AEGP_PluginID aegp_plugin_id, AEGP_MaskH maskH, PF_ParamIndex param_index, AEGP_StreamRefH *mask_opacity_streamPH); </pre>

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
<code>AEGP_GetStreamName</code>	<p>Get name of the stream (localized or forced English). is handle of <code>A_UTF16Char</code> (contains null terminated UTF16 string); must be disposed with <code>AEGP_FreeMemHandle</code>.</p> <pre> AEGP_GetStreamName(AEGP_PluginID pluginID, AEGP_StreamRefH streamH, A_Boolean force_englishB, AEGP_MemHandle *utf_stream_namePH); </pre> <p>NOTE: if <code>force_englishB</code> is TRUE, the default name will override any stream renaming which has been done (either programatically, or by the user).</p>
<code>AEGP_GetStreamUnitsText</code>	<p>Get stream units, formatted as text (localized or forced English); <code>unitsZ</code> up to <code>AEGP_MAX_STREAM_NAME_LEN + 1</code> long.</p> <pre> AEGP_GetStreamUnitsText(AEGP_StreamRefH streamH, A_Boolean force_englishB, A_char *unitsZ); </pre>
<code>AEGP_GetStreamProperties</code>	<p>Get stream's flags, as well as minimum and maximum values (as floats), if the stream <i>has</i> mins and maxes.</p> <p>StreamFlags values:</p> <pre> AEGP_StreamFlag_NONE AEGP_StreamFlag_HAS_MIN AEGP_StreamFlag_HAS_MAX </pre> <pre> AEGP_GetStreamProperties(AEGP_StreamRefH streamH, AEGP_StreamFlags *flagsP, A_FpLong *minP0, A_FpLong *maxP0); </pre>
<code>AEGP_IsStreamTimevarying</code>	<p>Returns whether or not the stream is affected by expressions.</p> <pre> AEGP_IsStreamTimevarying(AEGP_StreamRefH streamH, A_Boolean *is_timevaryPB); </pre>

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
AEGP_GetStreamType	<p>Get type (dimension) of a stream.</p> <pre> AEGP_GetStreamType(AEGP_StreamRefH streamH, AEGP_StreamType *stream_typeP); </pre> <p> 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 </p> <p>NOTE: always returns ThreeD_Spatial for position, regardless of whether or not the layer is 3D.</p>
AEGP_GetNewStreamValue	<p>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.</p> <pre> 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); </pre>
AEGP_DisposeStreamValue	<p>Dispose of stream value. Always deallocate values passed to the plug-in.</p> <pre> AEGP_DisposeStreamValue(AEGP_StreamValue2 *valueP); </pre>
AEGP_SetStreamValue	<p>Only legal when stream is not time-variant.</p> <pre> AEGP_SetStreamValue(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH streamH, AEGP_StreamValue2 *valueP); </pre>

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
<code>AEGP_GetLayerStreamValue</code>	<p>NOTE: This convenience function is only valid for streams with primitive data types, and not for <code>AEGP_ArbBlockVal</code>, <code>AEGP_MarkerValH</code> or <code>AEGP_MaskOutlineValH</code>. For these and other complex types, use <code>AEGP_GetNewStreamValue</code>, described above.</p> <pre> 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); </pre>
<code>AEGP_GetExpressionState</code>	<p>Determines whether expressions are enabled on the given <code>AEGP_StreamRefH</code>.</p> <pre> AEGP_GetExpressionState(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH streamH, A_Boolean *enabledPB); </pre>
<code>AEGP_SetExpressionState</code>	<p>Enables and disables expressions on the given <code>AEGP_StreamRefH</code>.</p> <pre> AEGP_SetExpressionState(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH streamH, A_Boolean enabledB); </pre>
<code>AEGP_GetExpression</code>	<p>Obtains the expression's text.</p> <pre> AEGP_GetExpression(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH streamH, AEGP_MemHandle *expressionHZ); </pre>
<code>AEGP_SetExpression</code>	<p>Sets the expression's text.</p> <pre> AEGP_SetExpression(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH streamH, const char* expressionP); </pre>

TABLE 73: AEGP_STREAMSUITE4

Function	Purpose
AEGP_DuplicateStreamRef	<p>Duplicates a given AEGP_StreamRefH. Dispose of the duplicate.</p> <pre> AEGP_DuplicateStreamRef(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH streamH, AEGP_StreamRefH *dup_streamPH); </pre>

MARKER STREAMS

New in CS3 (8.0), AEGP_MarkerSuite allows for better direct manipulation of marker data.

TABLE 74: AEGP_MARKERSUITE2

Function	Purpose
AEGP_NewMarker	<p>Creates a new marker.</p> <pre> AEGP_NewMarker(AEGP_MarkerValP *markerPP); </pre>
AEGP_DisposeMarker	<p>Disposes of a marker.</p> <pre> AEGP_DisposeMarker(AEGP_MarkerValP markerP); </pre>
AEGP_DuplicateMarker	<p>Duplicates a marker (didn't see <i>that</i> one coming, eh?).</p> <pre> AEGP_DuplicateMarker(AEGP_MarkerValP markerP, AEGP_MarkerValP *new_markerP); </pre>
AEGP_SetMarkerFlag	<p>Sets a marker flag's value.</p> <pre> AEGP_SetMarkerFlag(AEGP_MarkerValP markerP, AEGP_MarkerFlagType flagType, A_Boolean valueB); </pre> <p>Currently, AEGP_MarkerFlagType is one of the following:</p> <pre> AEGP_MarkerFlag_NONE AEGP_MarkerFlag_NAVIGATION </pre>
AEGP_GetMarkerFlag	<p>Gets the value (see above) of a given AEGP_MarkerFlagType.</p> <pre> AEGP_GetMarkerFlag(AEGP_ConstMarkerValP markerP, AEGP_MarkerFlagType flagType, A_Boolean *valueBP); </pre>

TABLE 74: AEGP_MARKERSUITE2

Function	Purpose
<code>AEGP_GetMarkerString</code>	Retrieves the UTF-16, NULL-terminated string located in the specified marker field. Must be disposed of by caller using <code>AEGP_FreeMemHandle</code> . <pre> AEGP_GetMarkerString(AEGP_PluginID id, AEGP_ConstMarkerValP markerP, AEGP_MarkerStringType strType, AEGP_MemHandle *unicodePH); </pre>
<code>AEGP_SetMarkerString</code>	Sets the specified field of a marker to the provided text. <pre> AEGP_SetMarkerString(AEGP_MarkerValP markerP, AEGP_MarkerStringType strType, const A_u_short *unicodeP, A_long lengthL); </pre>
<code>AEGP_CountCuePointParams</code>	Returns the number of cue point parameters. <pre> AEGP_CountCuePointParams(AEGP_ConstMarkerValP markerP, A_long *paramsLP); </pre>
<code>AEGP_GetIndCuePointParam</code>	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 <code>AEGP_FreeMemHandle</code> . <pre> AEGP_GetIndCuePointParam(AEGP_PluginID id, AEGP_ConstMarkerValP markerP, A_long param_indexL, AEGP_MemHandle *unicodeKeyPH, AEGP_MemHandle *uni_ValuePH); </pre>
<code>AEGP_SetIndCuePointParam</code>	Set the value of an indexed cue point parameter to the specified value. <code>key_lengthL</code> is “number of unicode characters”, and <code>value_lenL</code> is the length of the provided value. <code>unicode_KeyP</code> and <code>unicode_ValueP</code> point to UTF-16 data. <pre> 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); </pre>

TABLE 74: AEGP_MARKERSUITE2

Function	Purpose
AEGP_InsertCuePointParam	Inserts a cue point parameter. This call is following by AEGP_SetIndCuePointParam to actually set the data. <pre>AEGP_InsertCuePointParam(AEGP_MarkerValP markerP, A_long param_idxL);</pre>
AEGP_DeleteIndCuePointParam	Deletes the cue point param at the specified index. <pre>AEGP_DeleteIndCuePointParam(AEGP_MarkerValP markerP, A_long param_idxL);</pre>
AEGP_SetMarkerDuration	New in CS4. <pre>AEGP_SetMarkerDuration(AEGP_MarkerValP markerP, const A_Time *durationPT);</pre>
AEGP_GetMarkerDuration	New in CS4. <pre>AEGP_SetMarkerDuration(AEGP_ConstMarkerValP markerP, A_Time *durationPT);</pre>

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](#) calls to work with dynamic

streams. On the other hand, only those functions specific to dynamic streams are in this suite.

TABLE 75: 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 . <pre>AEGP_GetNewStreamRefForLayer(AEGP_PluginID aegp_plugin_id, AEGP_LayerH layerH, AEGP_StreamRefH *streamPH);</pre>
AEGP_GetNewStreamRefForMask	New for CS4. Retrieves the AEGP_StreamRefH corresponding to the mask. <pre>AEGP_GetNewStreamRefForMask(AEGP_PluginID aegp_plugin_id, AEGP_MaskRefH maskH, AEGP_StreamRefH *streamPH);</pre>
AEGP_GetStreamDepth	Retrieves the number of sub-streams associated with the given AEGP_StreamRefH. The initial layer has a depth of 0. <pre>AEGP_GetStreamDepth(AEGP_StreamRefH streamH, A_long *depthPL);</pre>
AEGP_GetStreamGroupingType	Retrieves the grouping type for the given AEGP_StreamRefH. <pre>AEGP_GetStreamGroupingType(AEGP_StreamRefH streamH, AEGP_StreamGroupingType *group_typeP);</pre> <p>AEGP_StreamGroupingType will be one of the following:</p> <pre>AEGP_StreamGroupingType_NONE AEGP_StreamGroupingType_LEAF AEGP_StreamGroupingType_NAMED_GROUP AEGP_StreamGroupingType_INDEXED_GROUP</pre>
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. <pre>AEGP_GetNumStreamsInGroup(AEGP_StreamRefH streamH, A_long *num_streamsPL);</pre>

TABLE 75: AEGP_DYNAMICSTREAMSUITE4

Function	Purpose
AEGP_GetDynamicStreamFlags	<p>Retrieves the flags for a given AEGP_StreamRefH.</p> <pre>AEGP_GetDynamicStreamFlags(AEGP_StreamRefH streamH, AEGP_DynStreamFlags *flagsP);</pre> <p>AEGP_DynStreamFlags will be one of the following:</p> <p>AEGP_DynStreamFlag_ACTIVE_EYEBALL AEGP_DynStreamFlag_HIDDEN AEGP_DynStreamFlag_DISABLED AEGP_DynStreamFlag_ELIDED</p> <p>AEGP_DynStreamFlag_ACTIVE_EYEBALL means that the stream is available for reading and writing.</p> <p>AEGP_DynStreamFlag_HIDDEN means that, while the stream is still read/write, it may not currently be visible in the UI.</p> <p>AEGP_DynStreamFlag_DISABLED means 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.</p> <p>AEGP_DynStreamFlag_ELIDED means 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.</p>
AEGP_SetDynamicStreamFlag	<p>Sets the specified flag for the AEGP_StreamRefH. Note; flags must be set individually. Undoable if undoableB is TRUE.</p> <pre>AEGP_SetDynamicStreamFlag(AEGP_StreamRefH streamH, AEGP_DynStreamFlags one_flag, A_Boolean undoableB, A_Boolean setB);</pre> <p>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.</p>

TABLE 75: AEGP_DYNAMICSTREAMSUITE4

Function	Purpose
AEGP_GetNewStreamRefByIndex	<p>Retrieves a sub-stream by index from a given AEGP_StreamRefH. Cannot be used on streams of type AEGP_StreamGroupingType_LEAF.</p> <pre> AEGP_GetNewStreamRefByIndex(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH parent_groupH, A_long indexL, AEGP_StreamRefH *streamPH); </pre>
AEGP_GetNewStreamRefByMatchname	<p>Retrieves a sub-stream by match name from a given AEGP_StreamRefH. Only legal for AEGP_StreamGroupingType_NAMED_GROUP.</p> <pre> AEGP_GetNewStreamRefByMatchname(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH parent_groupH, const A_char *match_nameZ, AEGP_StreamRefH *streamPH); </pre> <p>Here are some handy stream names, for which references may be retrieved:</p> <pre> 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 </pre>

TABLE 75: AEGP_DYNAMICSTREAMSUITE4

Function	Purpose
<code>AEGP_DeleteStream</code>	<p>Deletes the specified stream from a stream grouping. Note that the caller must still dispose of any <code>AEGP_StreamRefH</code> it's already acquired (allocated) via the API. Undoable. Only valid for children of type AEGP_StreamGroupingType_INDEXED_GROUP.</p> <pre>AEGP_DeleteStream(AEGP_StreamRefH streamH);</pre> <p>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.</p>
<code>AEGP_ReorderStream</code>	<p>Sets the new index of the specified <code>AEGP_StreamRefH</code>. Undoable. Only valid for children of <code>AEGP_StreamGroupingType_INDEXED_GROUP</code>. The <code>AEGP_StreamRefH</code> is updated to refer to the newly-ordered stream.</p> <pre>AEGP_ReorderStream(AEGP_StreamRefH streamH A_long new_indexL);</pre>
<code>AEGP_DuplicateStream</code>	<p>Duplicates the specified stream and appends it to the stream group. Undoable. Only valid for children of <code>AEGP_StreamGroupingType_INDEXED_GROUP</code>.</p> <pre>AEGP_DuplicateStream(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH streamH, A_long *new_indexPL0);</pre>
<code>AEGP_SetStreamName</code>	<p>Sets the name of the given <code>AEGP_StreamRefH</code>. Undoable. <code>nameZ</code> points to a null terminated UTF-16 string. Only valid for children of <code>AEGP_StreamGroupingType_INDEXED_GROUP</code>. NOTE: If you retrieve the name with <code>force_englishB</code> set to TRUE, you will get the canonical, UNchanged name of the stream.</p> <pre>AEGP_SetStreamName(AEGP_StreamRefH streamH, const A_UTF16Char *nameZ);</pre> <p>Note: Use this on an effect stream's group to change the display name of an effect.</p>

TABLE 75: AEGP_DYNAMICSTREAMSUITE4

Function	Purpose
AEGP_CanAddStream	<p>Returns whether or not it is currently possible to add a stream through the API.</p> <pre> AEGP_CanAddStream(AEGP_StreamRefH group_streamH, const A_char *match_nameZ, A_Boolean *can_addPB); </pre>
AEGP_AddStream	<p>Adds a stream to the specified stream group. Undoable. Only valid for AEGP_StreamGroupingType_INDEXED_GROUP.</p> <pre> AEGP_AddStream(AEGP_PluginID aegp_plugin_id, AEGP_StreamRefH indxd_grp_streamH, const A_char *match_nameZ, AEGP_StreamRefH *streamPH0); </pre>
AEGP_GetMatchName	<p>Retrieves the match name for the specified AEGP_StreamRefH. Note that this may differ from the display name, which can be retrieved using AEGP_GetStreamName, in AEGP_StreamSuite. nameZ can be up to AEGP_MAX_STREAM_MATCH_NAME_SIZE in length.</p> <pre> AEGP_GetMatchName(AEGP_StreamRefH streamH, A_char *nameZ); </pre>
AEGP_GetNewParentStreamRef	<p>Retrieves an AEGP_StreamRefH for the parent of the specified AEGP_StreamRefH.</p> <pre> AEGP_GetNewParentStreamRef(AEGP_PluginID plugin_id, AEGP_StreamRefH streamH, AEGP_StreamRefH *parentPH); </pre>
AEGP_GetStreamIsModified	<p>Returns whether or not the specified AEGP_StreamRefH has been modified. Note: the same result is available through the After Effect user interface by typing “UU” with the composition selected.</p> <pre> AEGP_GetStreamIsModified(AEGP_StreamRefH streamH, A_Boolean *modifiedPB); </pre>

TABLE 75: AEGP_DYNAMICSTREAMSUITE4

Function	Purpose
AEGP_GetStreamIndexInParent	<p>New in 7.0. Retrieves the index of a given stream, relative to its parent stream. Only valid for children of AEGP_StreamGroupingType_INDEXED_GROUP</p> <pre>AEGP_GetStreamIndexInParent(AEGP_StreamRefH streamH, A_long *indexPL);</pre> <p>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.</p>
AEGP_IsSeparationLeader	<p>New in CS4. 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.</p> <p>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.</p> <p>A Leader isn't always separated, call AEGP_AreDimensionsSeparated to find out if it is. As of CS4, the only stream that is ever separated is the layer's Position property. Please <i>*do not*</i> write code assuming that, we anticipate allowing separation of more streams in the future.</p> <pre>AEGP_IsSeparationLeader(AEGP_StreamRefH streamH, A_Boolean *leaderPB);</pre>
AEGP_AreDimensionsSeparated	<p>New in CS4. Methods such as AEGP_GetNewKeyframeValue that work on keyframe indices will most definitely <i>*not*</i> work on the Leader property, you will need to retrieve and operate on the Followers explicitly.</p> <pre>AEGP_AreDimensionsSeparated(AEGP_StreamRefH streamH, A_Boolean *separatedPB);</pre>
AEGP_SetDimensionsSeparated	<p>New in CS4. Valid only if AEGP_IsSeparationLeader() is true.</p> <pre>AEGP_AreDimensionsSeparated(AEGP_StreamRefH streamH, A_Boolean *separatedPB);</pre>

TABLE 75: AEGP_DYNAMICSTREAMSUITE4

Function	Purpose
AEGP_GetSeparationFollower	<p>New in CS4. Retrieve the Follower stream corresponding to a given dimension of the Leader stream. dimS can range from 0 to</p> <pre>AEGP_GetStreamValueDimensionality(leader_streamH) - 1.</pre> <pre>AEGP_GetSeparationFollower(AEGP_StreamRefH leader_streamH A_short dimS, AEGP_StreamRefH *follower_streamPH);</pre>
AEGP_IsSeparationFollower	<p>New in CS4. 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().</p> <pre>AEGP_IsSeparationFollower(AEGP_StreamRefH streamH A_Boolean *followerPB);</pre>
AEGP_GetSeparationLeader	<p>New in CS4. Valid on separation Followers only, returns the Leader it is part of.</p> <pre>AEGP_GetSeparationLeader(AEGP_StreamRefH follower_streamH, AEGP_StreamRefH *leader_streamPH);</pre>
AEGP_GetSeparationDimension	<p>New in CS4. Valid on separation Followers only, returns which dimension of the Leader it corresponds to.</p> <pre>AEGP_GetSeparationDimension(AEGP_StreamRefH follower_streamH, A_short *dimPS);</pre>

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 76: AEGP_KEYFRAMEsuite3

Function	Purpose
<code>AEGP_GetStreamNumKFs</code>	<p>Retrieves the number of keyframes on the given stream. Returns <code>AEGP_NumKF_NO_DATA</code> if the stream is not keyframe-able. Also, note that a stream without keyframes isn't necessarily constant; it can be altered by expressions.</p> <pre>AEGP_GetStreamNumKFs(AEGP_StreamRefH streamH, A_long *num_kfsPL);</pre>
<code>AEGP_GetKeyframeTime</code>	<p>Retrieves the time of the specified keyframe.</p> <pre>AEGP_GetKeyframeTime(AEGP_StreamRefH streamH, AEGP_KeyframeIndex index, AEGP_LTimeMode time_mode, A_Time *timePT);</pre>
<code>AEGP_InsertKeyframe</code>	<p>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.</p> <pre>AEGP_InsertKeyframe(AEGP_StreamRefH streamH, AEGP_LTimeMode time_mode, const A_Time *timePT, AEGP_KeyframeIndex *key_indexP);</pre>
<code>AEGP_DeleteKeyframe</code>	<p>Deletes the specified keyframe.</p> <pre>AEGP_DeleteKeyframe(AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index);</pre>

TABLE 76: AEGP_KEYFRAMESUITE3

Function	Purpose
AEGP_GetNewKeyframeValue	<p>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.</p> <pre> AEGP_GetNewKeyframeValue(AEGP_PluginID plugin_id, AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, AEGP_StreamValue2 *valueP); </pre>
AEGP_SetKeyframeValue	<p>Sets the stream's value at the time of the keyframe.</p> <pre> AEGP_SetKeyframeValue(AEGP_StreamRefH streamH, AEGP_KeyframeIndex index, const AEGP_StreamValue2 *valP); </pre>
AEGP_GetStreamValue Dimensionality	<p>Retrieves the dimensionality of the stream's value.</p> <pre> AEGP_GetStreamValueDimensionality(AEGP_StreamRefH streamH, A_short *value_dimPS); </pre>
AEGP_GetStreamTemporal Dimensionality	<p>Retrieves the temporal dimensionality of the stream.</p> <pre> AEGP_GetStreamTemporalDimensionality(AEGP_StreamRefH streamH, A_short *t_dimPS); </pre>
AEGP_GetNewKeyframe SpatialTangents	<p>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.</p> <pre> AEGP_GetNewKeyframeSpatialTangents(AEGP_PluginID plugin_id, AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, AEGP_StreamValue2 *in_tanP0, AEGP_StreamValue2 *out_tanP0); </pre>

TABLE 76: AEGP_KEYFRAMEsuite3

Function	Purpose
<code>AEGP_SetKeyframeSpatialTangents</code>	<p>Specifies the tangential <code>AEGP_StreamValue2s</code> to be used for the stream's value at the time of the keyframe. The <code>AEGP_StreamValue2s</code> passed for in and out tangents are not adopted by After Effects, and must be disposed of using <code>AEGP_DisposeStreamValue</code>.</p> <pre> AEGP_SetKeyframeSpatialTangents(AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, const AEGP_StreamValue2*in_tP0, const AEGP_StreamValue2*out_tP0); </pre> <p>NOTE: In <code>AEGP_KeyframeSuite2</code> 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.</p>
<code>AEGP_GetKeyframeTemporalEase</code>	<p>Retrieves the <code>AEGP_KeyframeEases</code> associated with the specified dimension of the stream's value at the time of the keyframe. <code>dimensionL</code> ranges from 0 to <code>(temporal_dimensionality - 1)</code>.</p> <pre> AEGP_GetKeyframeTemporalEase(AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, A_long dimensionL, AEGP_KeyframeEase *in_easeP0, AEGP_KeyframeEase *out_easeP0); </pre> <p>NOTE: the returned ease values must be multiplied by layer height to match the values displayed in the After Effects UI.</p>
<code>AEGP_SetKeyframeTemporalEase</code>	<p>Specifies the <code>AEGP_KeyframeEases</code> to be used for the stream's value at the time of the keyframe. <code>dimensionL</code> ranges from 0 to <code>(temporal_dimensionality - 1)</code>. The <code>AEGP_KeyframeEases</code> passed are not adopted by After Effects.</p> <pre> AEGP_SetKeyframeTemporalEase(AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, A_long dimL, const AEGP_KeyframeEase *in_P0, const AEGP_KeyframeEase *outP0); </pre>

TABLE 76: AEGP_KEYFRAMESUITE3

Function	Purpose
AEGP_GetKeyframeFlags	<p>Retrieves the flags currently set for the keyframe.</p> <pre> AEGP_GetKeyframeFlags(AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, AEGP_KeyframeFlags *flagsP); </pre> <p>*flagsP will be a combination of the following:</p> <pre> AEGP_KeyframeFlag_NONE AEGP_KeyframeFlag_TEMPORAL_CONTINUOUS AEGP_KeyframeFlag_TEMPORAL_AUTOBEZIER AEGP_KeyframeFlag_SPATIAL_CONTINUOUS AEGP_KeyframeFlag_SPATIAL_AUTOBEZIER AEGP_KeyframeFlag_ROVING </pre>
AEGP_SetKeyframeFlag	<p>Sets the specified flag for the keyframe. Flags must be set individually.</p> <pre> AEGP_SetKeyframeFlag(AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, AEGP_KeyframeFlags flag, A_Boolean valueB); </pre>
AEGP_GetKeyframeInterpolation	<p>Retrieves the in and out AEGP_KeyframeInterpolationTypes for the specified keyframe.</p> <pre> AEGP_GetKeyframeInterpolation(AEGP_StreamRefH streamH, AEGP_KeyframeIndex key_index, AEGP_KeyframeInterpolationType *inP0, AEGP_KeyframeInterpolationType *outP0); </pre> <p>AEGP_KeyframeInterpolationType is one of the following:</p> <pre> AEGP_KeyInterp_NONE AEGP_KeyInterp_LINEAR AEGP_KeyInterp_BEZIER AEGP_KeyInterp_HOLD </pre>

TABLE 76: AEGP_KEYFRAMEsuite3

Function	Purpose
<code>AEGP_SetKeyframeInterpolation</code>	<p>Specifies the in and out <code>AEGP_KeyframeInterpolationTypes</code> to be used for the given keyframe.</p> <pre> AEGP_SetKeyframeInterpolation(AEGP_StreamRefHstreamH, AEGP_KeyframeIndex key_index, AEGP_KeyframeInterpolationType in_interp, AEGP_KeyframeInterpolationType out_interp); </pre>
<code>AEGP_StartAddKeyframes</code>	<p>Informs After Effects that you're going to be adding several keyframes to the specified stream. After Effects will return an allocated opaque <code>AEGP_AddKeyframesInfoH</code>, for use with the calls below.</p> <pre> AEGP_StartAddKeyframes(AEGP_StreamRefH streamH, AEGP_AddKeyframesInfoH*akPH); </pre>
<code>AEGP_AddKeyframes</code>	<p>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.</p> <pre> AEGP_AddKeyframes(AEGP_AddKeyframesInfoH akH, AEGP_LTimeMode time_mode, const A_Time *timePT, A_long *indexPL); </pre>
<code>AEGP_SetAddKeyframe</code>	<p>Sets the value of the specified keyframe.</p> <pre> AEGP_SetAddKeyframe(AEGP_AddKeyframesInfoH akH, A_long indexL, const AEGP_StreamValue2*valueP); </pre>
<code>AEGP_EndAddKeyframes</code>	<p>Tells After Effects you're done adding keyframes.</p> <pre> AEGP_EndAddKeyframes(A_Boolean addB, AEGP_AddKeyframesInfoH akH); </pre>

ADDING MULTIPLE KEYFRAMES

Each time you call [AEGP_InsertKeyframe\(\)](#), 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](#), passing it an opaque `AEGP_AddKeyframesInfoH`. For each keyframe to add, call [AEGP_AddKeyframes](#) to set the time to be used (and get the newly-added keyframe's index), then [AEGP_SetAddKeyframe](#) to specify the value to be used. Once you're finished, call [AEGP_EndAddKeyframes](#) to let know After Effects know it's time to add the changed parameter stream to the undo stack.

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](#), above, and passing `AEGP_StreamType_TEXT_DOCUMENT` as the `AEGP_StreamType`.

TABLE 77: AEGP_TEXTDOCUMENTSUITE1

Function	Purpose
<code>AEGP_GetNewText</code>	Retrieves the UTF-16, NULL-terminated string used in the <code>AEGP_TextDocumentH</code> . Note: After Effects will allocate the <code>AEGP_MemHandle</code> ; your plug-in must dispose of it when done using <code>AEGP_FreeMemHandle</code> . <code>AEGP_GetNewText(</code> <code>AEGP_PluginID</code> <code>id,</code> <code>AEGP_TextDocumentH</code> <code>text_docH,</code> <code>AEGP_MemHandle</code> <code>*unicodePH);</code>
<code>AEGP_SetText</code>	Specifies the text to be used by the <code>AEGP_TextDocumentH</code> . <code>AEGP_SetText(</code> <code>AEGP_TextDocumentH</code> <code>text_docH,</code> <code>const A_u_short</code> <code>*unicodePS,</code> <code>long</code> <code>lengthL);</code>

WORKING WITH TEXT OUTLINES

The `AEGP_TextLayerSuite` provides access to the actual outlines of the text used by text layers. Once you have a path, you can manipulate it with [PF_PathQuerySuite](#) and [PF_PathDataSuite](#).

TABLE 78: AEGP_TEXTLAYERSUITE1

Function	Purpose
<code>AEGP_GetNewTextOutlines</code>	Allocates and returns a handle to the <code>AEGP_TextOutlinesHs</code> associated with the specified layer. <code>outlinesPH</code> will be <code>NULL</code> if there are no <code>AEGP_TextOutlinesHs</code> associated with <code>layerH</code> (in other words, if it's not a text layer). <pre>AEGP_GetNewTextOutlines(AEGP_LayerH layerH, / const A_Time *layer_timePT, AEGP_TextOutlinesH *outlinesPH);</pre>
<code>AEGP_DisposeTextOutlines</code>	Dispose of those outlines we allocated on your behalf! <pre>AEGP_DisposeTextOutlines(AEGP_TextOutlinesH outlinesH);</pre>
<code>AEGP_GetNumTextOutlines</code>	Retrieves the number of text outlines for the layer. <pre>AEGP_GetNumTextOutlines(AEGP_TextOutlinesH outlinesH, A_long *num_otlnsPL);</pre>
<code>AEGP_GetIndexedTextOutline</code>	Returns a <code>PF_PathOutlinePtr</code> for the specified text outline. <pre>AEGP_GetIndexedTextOutline(AEGP_TextOutlinesH outlinesH, A_long path_indexL, PF_PathOutlinePtr *pathPP);</pre>

RENDER SUITE

Since we introduced the AEGP API, we've been asked to provide functions for retrieving rendered frames. This function suite allows you to do just that.

TABLE 79: AEGP_RENDERSUITE2

Function	Purpose
<code>AEGP_RenderAndCheckoutFrame</code>	<p>Retrieves an <code>AEGP_FrameReceiptH</code> (not the actual pixels) for the frame requested. 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.</p> <pre>AEGP_RenderAndCheckoutFrame(AEGP_RenderOptionsH optionsH, AEGP_RenderSuiteCheckForCancel cancel_functionP0, AEGP_CancelRefcon cancel_function_refconP0, AEGP_FrameReceiptH *receiptPH);</pre>
<code>AEGP_CheckinFrame</code>	<p>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!</p> <pre>AEGP_CheckinFrame(AEGP_FrameReceiptH receiptH);</pre>
<code>AEGP_GetReceiptWorld</code>	<p>Retrieves the pixels (<code>AEGP_WorldH</code>) associated with the referenced <code>AEGP_FrameReceiptH</code>.</p> <pre>AEGP_GetReceiptWorld(AEGP_FrameReceiptH receiptH, AEGP_WorldH *worldPH);</pre>
<code>AEGP_GetRenderedRegion</code>	<p>Retrieves an <code>A_LRect</code> containing the region of the <code>AEGP_FrameReceiptH</code>'s <code>AEGP_WorldH</code> that has already been rendered. Remember that, in 6.0 and later versions, it is increasingly 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.</p> <pre>AEGP_GetRenderedRegion(AEGP_FrameReceiptH receiptH, A_LRect *regionP);</pre>

TABLE 79: AEGP_RENDERSUITE2

Function	Purpose
<code>AEGP_IsRenderedFrameSufficient</code>	<p>Given two sets of <code>AEGP_RenderOptionsH</code>, After Effects will return TRUE if the already-rendered pixels are still valid for the proposed <code>AEGP_RenderOptionsH</code>.</p> <pre> AEGP_IsRenderedFrameSufficient(AEGP_RenderOptionsH rendered_optionsH, AEGP_RenderOptionsH proposed_optionsH, A_Boolean *is_sufficientPB); </pre>
<code>AEGP_RenderNewItemSoundData</code>	<p>Obtains an <code>AEGP_ItemH</code>'s audio at the given time, of the given duration, in the given format. The plug-in must dispose of the returned <code>AEGP_SoundDataH</code> (which may be NULL if no audio is available).</p> <pre> AEGP_RenderNewItemSoundData(AEGP_ItemH itemH, const A_Time *start_timePT, const A_Time *durationPT, const AEGP_SoundDataFormat *formatP, AEGP_SoundDataH *new_dataPH); </pre> <p>NOTE: This function, if called as part of <code>AEGP_ItemSuite2</code>, provides a render interruptible using mouse clicks, unlike the version published here in <code>AEGP_RenderSuite</code>.</p>
<code>AEGP_GetCurrentTimestamp</code>	<p>Retrieves the current <code>AEGP_TimeStamp</code> of the project. The <code>AEGP_TimeStamp</code> is updated whenever an item is touched in a way that affects rendering.</p> <pre> AEGP_GetCurrentTimestamp(AEGP_TimeStamp *time_stampP); </pre>
<code>AEGP_HasItemChangedSinceTimestamp</code>	<p>Returns whether the video of an <code>AEGP_ItemH</code> has changed since the given <code>AEGP_TimeStamp</code>. Note: this does not track changes in audio.</p> <pre> AEGP_HasItemChangedSinceTimestamp(AEGP_ItemH itemH, const A_Time *start_timeP, const A_Time *durationP, const AEGP_TimeStamp *time_stampP, A_Boolean *changedPB); </pre>

TABLE 79: AEGP_RENDERSUITE2

Function	Purpose
<code>AEGP_IsItemWorthwhileToRender</code>	<p>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() and checking in. This function is to be used with AEGP_HasItemChangedSinceTimestamp(), not alone.</p> <pre> AEGP_IsItemWorthwhileToRender(AEGP_RenderOptionsH roH, const AEGP_TimeStamp *time_stampP, A_Boolean *worthwhilePB); </pre>
<code>AEGP_CheckinRenderedFrame</code>	<p>Provide a rendered frame (<code>AEGP_PlatformWorldH</code>) to After Effects, which adopts it. <code>ticksL</code> is the approximate time required to render the frame.</p> <pre> AEGP_CheckinRenderedFrame(AEGP_RenderOptionsH roH, const AEGP_TimeStamp* time_stampP, A_u_long ticksL, AEGP_PlatformWorldH imageH); </pre>

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](#) 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 80: AEGP_EFFECTSUITE3

Function	Purpose
<code>AEGP_GetLayerNumEffects</code>	<p>Get number of effects applied to a layer.</p> <pre> AEGP_GetLayerNumEffects(AEGP_LayerH layerH, A_long *num_effectsPL); </pre>

TABLE 80: AEGP_EFFECTSUITE3

Function	Purpose
<code>AEGP_GetLayerEffectByIndex</code>	<p>Retrieves (by index) a reference to an effect applied to the layer.</p> <pre> AEGP_GetLayerEffectByIndex(AEGP_PluginID aegp_plugin_id, AEGP_LayerH layerH, AEGP_EffectIndex effect_indexL, AEGP_EffectRefH *effectPH); </pre>
<code>AEGP_GetInstalledKeyFromLayerEffect</code>	<p>Given an <code>AEGP_EffectRefH</code>, retrieves its associated <code>AEGP_InstalledEffectKey</code>.</p> <pre> AEGP_GetInstalledKeyFromLayerEffect(AEGP_EffectRefH effect_refH, AEGP_InstalledEffectKey *keyP); </pre>
<code>AEGP_GetEffectParamUnionByIndex</code>	<p>Returns description of effect parameter. Do not use the value(s) in the <code>ParamDef</code> returned by this function (Use AEGP_GetNewStreamValue() instead); it's provided so AEGPs can access parameter defaults, checkbox names, and pop-up strings.</p> <p>Use AEGP_GetEffectNumParamStreams() from the <code>StreamSuite</code> to get the stream count, useful for determining the maximum <code>param_index</code>. The last parameter is optional;</p> <pre> AEGP_GetEffectParamUnionByIndex(AEGP_PluginID aegp_plugin_id, AEGP_EffectRefH effectH, PF_ParamIndex param_index, PF_ParamType *param_typeP PF_ParamDefUnion *uP0); </pre>
<code>AEGP_GetEffectFlags</code>	<p>Obtains the flags for the given <code>AEGP_EffectRefH</code>.</p> <pre> AEGP_GetEffectFlags(AEGP_EffectRefH effect_refH, AEGP_EffectFlags *effect_flagsP); </pre> <p>Flags will be a combination of the following:</p> <pre> AEGP_EffectFlags_NONE AEGP_EffectFlags_ACTIVE AEGP_EffectFlags_AUDIO_ONLY AEGP_EffectFlags_AUDIO_TOO AEGP_EffectFlags_MISSING </pre>

TABLE 80: AEGP_EFFECTSUITE3

Function	Purpose
<code>AEGP_SetEffectFlags</code>	<p>Sets the flags (enumerated above) for the given <code>AEGP_EffectRefH</code>, masked by a different set of effect flags.</p> <pre> AEGP_SetEffectFlags(AEGP_EffectRefH effect_refH, AEGP_EffectFlags mask, AEGP_EffectFlags effect_flags); </pre>
<code>AEGP_ReorderEffect</code>	<p>Change the order of applied effects (pass the requested index).</p> <pre> AEGP_ReorderEffect(AEGP_EffectRefH effect_refH, A_long effect_indexL); </pre>
<code>AEGP_EffectCallGeneric</code>	<p>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. As of 7.0, pass PF_Cmd_COMPLETELY_GENERAL for <code>effect_cmd</code>.</p> <pre> AEGP_EffectCallGeneric(AEGP_PluginID aegp_plugin_id, AEGP_EffectRefH effectH, const A_Time *timePT, PF_Cmd effect_cmd, void *extraPV); </pre>
<code>AEGP_DisposeEffect</code>	<p>Disposes of an <code>AEGP_EffectRefH</code>. Use this to dispose of any <code>AEGP_EffectRefH</code> returned by After Effects.</p> <pre> AEGP_DisposeEffect(AEGP_EffectRefH effectH); </pre>
<code>AEGP_ApplyEffect</code>	<p>Apply an effect to a given layer. Returns the newly-created <code>AEGP_EffectRefH</code>.</p> <pre> AEGP_ApplyEffect(AEGP_PluginID aegp_plugin_id, AEGP_LayerH layerH, AEGP_InstalledEffectKey key, AEGP_EffectRefH *effect_refPH); </pre>
<code>AEGP_DeleteLayerEffect</code>	<p>Remove an applied effect.</p> <pre> AEGP_DeleteLayerEffect(AEGP_EffectRefH effect_refH); </pre>
<code>AEGP_GetNumInstalledEffects</code>	<p>Returns the count of effects installed in After Effects.</p> <pre> AEGP_GetNumInstalledEffects(A_long *num_installed_effectsPL); </pre>

TABLE 80: AEGP_EFFECTSUITE3

Function	Purpose
<code>AEGP_GetNextInstalledEffect</code>	<p>Returns the <code>AEGP_InstalledEffectKey</code> of the next installed effect. Pass <code>AEGP_InstalledEffectKey_NONE</code> as the first parameter to obtain the first <code>AEGP_InstalledEffectKey</code>.</p> <pre> AEGP_GetNextInstalledEffect(AEGP_InstalledEffectKey key, AEGP_InstalledEffectKey *next_keyPH); </pre>
<code>AEGP_GetEffectName</code>	<p>Get name of the effect. <code>nameZ</code> can be up to <code>AEGP_MAX_EFFECT_NAME_SIZE + 1</code> long.</p> <pre> AEGP_GetEffectName(AEGP_InstalledEffectKey installed_key, A_char *nameZ); </pre> <p>Note: use AEGP_SetStreamName to change the display name of an effect.</p>
<code>AEGP_GetEffectMatchName</code>	<p>Get match name of an effect (defined in PiPL). <code>match_nameZ</code> up to <code>AEGP_MAX_EFFECT_MATCH_NAME_SIZE + 1</code> long.</p> <pre> AEGP_GetEffectMatchName(AEGP_EffectRefH effectH, A_char *match_nameZ); </pre> <p>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.</p>
<code>AEGP_GetEffectCategory</code>	<p>Menu category of effect. <code>categoryZ</code> can be up to <code>AEGP_MAX_EFFECT_CATEGORY_NAME_SIZE + 1</code> long.</p> <pre> AEGP_GetEffectCategory(AEGP_EffectRefH effectH, A_char *categoryZ); </pre>
<code>AEGP_DuplicateEffect</code>	<p>Duplicates a given <code>AEGP_EffectRefH</code>. Caller must dispose of duplicate when finished.</p> <pre> AEGP_DuplicateEffect(AEGP_EffectRefH orig_effect_refH, AEGP_EffectRefH *dupe_refPH); </pre>

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](#) then immediately [remove](#) an effect, the layer's effect controls will be made visible. Tricky, eh?

STREAMREFS AND EFFECTREFS

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](#).

REGISTERING WITH AFTER EFFECTS

Register functions for After Effects' use.

TABLE 81: AEGP_REGISTERSUITE5

Function	Purpose
<code>AEGP_RegisterCommandHook</code>	<p>Register a hook (command handler) function with After Effects. If you are replacing a function which After Effects also handles, <code>AEGP_HookPriority</code> determines whether your plug-in gets it first.</p> <p><code>AEGP_HP_BeforeAE</code> <code>AEGP_HP_AfterAE</code></p> <p>For each menu item you add, obtain your own <code>AEGP_Command</code> using AEGP_GetUniqueCommand() prior registering a single <code>command_hook_func</code>. Determine which command was sent within this hook function, and act accordingly.</p> <p>Currently, <code>AEGP_HookPriority</code> is ignored.</p> <pre>AEGP_RegisterCommandHook(AEGP_PluginID aegp_plugin_id, AEGP_HookPriority hook_priority, AEGP_Command command, AEGP_CommandHook command_hook_func void *refconPV);</pre>

TABLE 81: AEGP_REGISTERSUITE5

Function	Purpose
AEGP_RegisterUpdateMenu Hook	<p>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.</p> <pre>AEGP_RegisterUpdateMenuHook(AEGP_PluginID aegp_plugin_id, AEGP_UpdateMenuHook update_menu_hook_func, void *refconPV);</pre>
AEGP_RegisterDeathHook	<p>Register your termination function. Called when the application quits.</p> <pre>AEGP_RegisterDeathHook(AEGP_PluginID aegp_plugin_id, AEGP_DeathHook death_hook_func, void *refconPV);</pre>
AEGP_RegisterVersionHook	Currently not called.
AEGP_RegisterAboutString Hook	Currently not called.
AEGP_RegisterAboutHook	Currently not called.
AEGP_RegisterArtisan	<p>Register your Artisan. See the Artisan chapter for more details.</p> <pre>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);</pre>
AEGP_RegisterIO	<p>Register your AEIO plug-in. See the AEIO section for more details.</p> <pre>AEGP_RegisterIO (AEGP_PluginID aegp_plugin_id, AEGP_IORefcon aegp_refconP, const AEIO_ModuleInfo *io_infoP, const AEIO_FunctionBlock4 *aeio_fcn_blockP);</pre>
AEGP_RegisterIdleHook	<p>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).</p> <pre>AEGP_RegisterIdleHook(AEGP_PluginID aegp_plugin_id, AEGP_IdleHook idle_hook_func, AEGP_IdleRefcon refconP);</pre>

TABLE 81: AEGP_REGISTERSUITE5

Function	Purpose
<code>AEGP_RegisterInteractiveArtisan</code>	Registers your AEGP as an interactive artisan, for use in previewing and rendering all layers in a given composition. <pre>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);</pre>
<code>AEGP_RegisterPresetLocalizationString</code>	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). <pre>AEGP_RegisterPresetLocalizationString(const A_char *english_nameZ, const A_char *localized_nameZ);</pre>

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](#) to assign handlers to menu events.

TABLE 82: AEGP_COMMANDSUITE1

Function	Purpose
<code>AEGP_GetUniqueCommand</code>	Obtain a unique command identifier. Use the Register Suite to register a handler for the command. <pre>AEGP_GetUniqueCommand(AEGP_Command *unique_commandP);</pre> <p>Note: On occasion After Effects will send command 0 (zero), so don't use that as part of your command handling logic.</p>

TABLE 82: AEGP_COMMANDSUITE1

Function	Purpose
AEGP_InsertMenuCommand	<p>Add menu command. Menu_ID can be:</p> <p>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</p> <p>Locations can be set to a specific location in the menu or can be one assigned by After Effects:</p> <p>AEGP_MENU_INSERT_SORTED AEGP_MENU_INSERT_AT_BOTTOM AEGP_MENU_INSERT_AT_TOP</p> <p>Note that in CS5 and later, the BOTTOM and TOP options are no longer supported for AEGP_Menu_WINDOW, and will return an error.</p> <pre>AEGP_InsertMenuCommand(AEGP_Command command, const A_char *nameZ, AEGP_MenuID menu_id, A_long after_itemL);</pre>
AEGP_RemoveMenuCommand	<p>Remove a menu command. If you were so motivated, you could remove ALL of the After Effects menu items.</p> <pre>AEGP_RemoveMenuCommand(AEGP_Command command);</pre>
AEGP_SetCommandName	<p>Set menu name of a command.</p> <pre>AEGP_SetCommandName(AEGP_Command command, const A_char *nameZ);</pre>
AEGP_EnableCommand	<p>Enable a menu command.</p> <pre>AEGP_EnableCommand(AEGP_Command command);</pre>

TABLE 82: AEGP_COMMANDSUITE1

Function	Purpose
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, contact API Engineering for the command number. 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

UTILITY FUNCTIONS

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 83: AEGP_UTILITYSUITE5

Function	Purpose
AEGP_ReportInfo	Displays dialog with name of the AEGP followed by the string passed. AEGP_ReportInfo(AEGP_PluginID aegp_plugin_id, A_char *info_stringZ);

TABLE 83: AEGP_UTILITYSUITE5

Function	Purpose
AEGP_GetDriverSpecVersion	Returns version of AEGPDriver plug-in (use to determine supported features). <pre>AEGP_GetDriverSpecVersion(A_short *major_versionPS, A_short *minor_versionPS);</pre>
AEGP_StartQuietErrors	Silences errors. Must be balanced with AEGP_EndQuietErrors. The AEGP_ErrReportState is an opaque structure private to After Effects. <pre>AEGP_StartQuietErrors(AEGP_ErrReportState *err_stateP);</pre>
AEGP_EndQuietErrors	Re-enables errors. <pre>AEGP_EndQuietErrors(AEGP_ErrReportState *err_stateP)</pre>
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. <pre>AEGP_StartUndoGroup(const A_char *undo_nameZ);</pre>
AEGP_EndUndoGroup	Ends the undo list. <pre>AEGP_EndUndoGroup();</pre>
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. <pre>AEGP_RegisterWithAEGP(AEGP_GlobalRefcon global_refcon, const A_char *plugin_nameZ, AEGP_PluginID *plugin_id);</pre>
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. <pre>AEGP_GetMainHWND(void *main_hwnd);</pre>

TABLE 83: AEGP_UTILITYSUITE5

Function	Purpose
AEGP_ShowHideAllFloaters	<p>Toggles whether or not floating palettes are displayed. Use this with care; users get twitchy when you unexpectedly change the UI on them.</p> <pre>AEGP_ShowHideAllFloaters(A_Boolean include_tool_palB);</pre> <p>NOTE: As of 7.0, this will no longer work on non-ADM palettes; document windows and floating palettes receive the same treatment.</p>
AEGP_PaintPalGetForeColor	<p>Retrieves the foreground color from the paint palette.</p> <pre>AEGP_PaintPalGetForeColor(AEGP_ColorVal *fore_colorP);</pre>
AEGP_PaintPalGetBackColor	<p>Retrieves the background color from the paint palette.</p> <pre>AEGP_PaintPalGetBackColor(AEGP_ColorVal *back_colorP);</pre>
AEGP_PaintPalSetForeColor	<p>Sets the foreground color in the paint palette.</p> <pre>AEGP_PaintPalSetForeColor(const AEGP_ColorVal *fore_colorP);</pre>
AEGP_PaintPalSetBackColor	<p>Sets the background color in the paint palette.</p> <pre>AEGP_PaintPalSetBackColor(const AEGP_ColorVal *back_colorP);</pre>
AEGP_CharPalGetFillColor	<p>Retrieves the fill color from the character palette.</p> <pre>AEGP_CharPalGetFillColor(A_Boolean *is_fcolor_definedPB, AEGP_ColorVal *fill_colorP);</pre>
AEGP_CharPalGetStrokeColor	<p>Retrieves the stroke color from the character palette.</p> <pre>AEGP_CharPalGetStrokeColor(A_Boolean *is_scolor_definedPB, AEGP_ColorVal *stroke_colorP);</pre>
AEGP_CharPalSetFillColor	<p>Sets the fill color in the character palette.</p> <pre>AEGP_CharPalSetFillColor(const AEGP_ColorVal *fill_colorP);</pre>
AEGP_CharPalSetStrokeColor	<p>Sets the stroke color in the character palette.</p> <pre>AEGP_CharPalSetStrokeColor(const AEGP_ColorVal *stroke_colorP);</pre>

TABLE 83: AEGP_UTILITYSUITE5

Function	Purpose
<code>AEGP_CharPalIsFillColorUIFrontmost</code>	Returns whether or not the fill color is frontmost. If it isn't, the stroke color is frontmost. <pre>AEGP_CharPalIsFillColorUIFrontmost(A_Boolean *is_fcolor_selectedPB);</pre>
<code>AEGP_ConvertFpLongToHSFRatio</code>	Returns an <code>A_Ratio</code> interpretation of the given <code>A_FpLong</code> . Useful for horizontal scale factor interpretation. <pre>AEGP_ConvertFpLongToHSFRatio(A_FpLong numberF, A_Ratio *ratioPR);</pre>
<code>AEGP_ConvertHSFRatioToFpLong</code>	Returns an <code>A_FpLong</code> interpretation of the given <code>A_Ratio</code> . Useful for horizontal scale factor interpretation. <pre>AEGP_ConvertHSFRatioToFpLong(A_Ratio ratioR, A_FpLong *numberPF);</pre>
<code>AEGP_CauseIdleRoutinesToBeCalled</code>	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. <pre>AEGP_CauseIdleRoutinesToBeCalled(void);</pre>
<code>AEGP_GetSuppressInteractiveUI</code>	Returns whether After Effects is running without a user interface. <pre>AEGP_GetSuppressInteractiveUI(A_Boolean *ui_is_suppressedPB);</pre>
<code>AEGP_WriteToOSConsole</code>	Sends a string to the OS console. <pre>AEGP_WriteToOSConsole(const A_char *textZ);</pre>
<code>AEGP_WriteToDebugLog</code>	Writes a message to the debug log, or to the OS command line if After Effects was launched with the “-debug” option. <pre>AEGP_WriteToDebugLog(const A_char *subsystemZ, const A_char *event_typeZ, const A_char *infoZ);</pre>
<code>AEGP_GetLastErrorMessage</code>	New in 7.0. 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. <pre>AEGP_GetLastErrorMessage(A_long buffer_size, A_char *error_string, A_Err *error_num);</pre>

TABLE 83: AEGP_UTILITYSUITE5

Function	Purpose
<code>AEGP_IsScriptingAvailable</code>	New in 7.0. Returns TRUE if scripting is available to the plug-in. <pre>AEGP_IsScriptingAvailable(A_Boolean *outAvailablePB);</pre>
<code>AEGP_ExecuteScript</code>	New in 7.0. Have After Effects execute a script. The script passed in can be in either UTF-8 or the current application encoding (if <code>platform_encodingB</code> 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 <code>outResultPH0</code> . <pre>AEGP_ExecuteScript(AEGP_PluginID inPlugin_id, const A_char *inScriptZ, const A_Boolean platform_encodingB, AEGP_MemHandle *outResultPH0, AEGP_MemHandle *outErrStringPH0);</pre>
<code>AEGP_HostIsActivated</code>	New in 7.0. Returns TRUE if the user has successfully activated After Effects. <pre>AEGP_HostIsActivated(A_Boolean *is_activatedPB);</pre>
<code>AEGP_GetPluginPlatformRef</code>	New in 7.0. On Mac OS, returns a <code>CFBundleRef</code> 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 <code>DLLInstance</code>). <pre>AEGP_GetPluginPlatformRef(AEGP_PluginID plug_id, void **plat_refPPV);</pre>
<code>AEGP_UpdateFontList</code>	New in CS3 (8.0). Rescans the system font list. <pre>AEGP_UpdateFontList();</pre>

HANDLING HANDLES

Manages memory used by the AEGP. Whenever memory related errors are encountered, After Effects can report errors for you. `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 84: AEGP_MEMORYSUITE1

Function	Purpose
AEGP_NewMemHandle	<p>Create a new memory handle. This memory is guaranteed to be 16-byte aligned. plugin_id is the number you obtained from AEGP_RegisterWithAEGP(). Use what_Z to identify the memory you are asking for. After Effects uses the string to display any related error messages.</p> <pre>AEGP_NewMemHandle(AEGP_PluginID *plugin_id, const A_char *whatZ, AEGP_MemSize size, AEGP_MemFlag flags, AEGP_MemHandle *memPH);</pre>
AEGP_FreeMemHandle	<p>Release a handle you allocated using AEGP_NewMemHandle().</p> <pre>AEGP_FreeMemHandle(AEGP_MemHandle memH);</pre>
AEGP_LockMemHandle	<p>Locks the handle into memory (cannot be moved by OS). Use this function prior to using memory allocated by AEGP_NewMemHandle. Can be nested.</p> <pre>AEGP_LockMemHandle(AEGP_MemHandle memH, void **ptr_to_ptr);</pre>
AEGP_UnlockMemHandle	<p>Allows OS to move the referenced memory. Always balance lock calls with unlocks.</p> <pre>AEGP_UnlockMemHandle(AEGP_MemHandle memH);</pre>
AEGP_GetMemHandleSize	<p>Returns the allocated size of the handle.</p> <pre>AEGP_GetMemHandleSize AEGP_MemHandle memH, AEGP_MemSize *sizeP);</pre>
AEGP_ResizeMemHandle	<p>Changes the allocated size of the handle.</p> <pre>AEGP_ResizeMemHandle(const char *whatZ, AEGP_MemSize new_size, AEGP_MemHandle memH);</pre>
AEGP_SetMemReportingOn	<p>If After Effects runs into problems with the memory handling, the error should be reported to the user. Make use of this during development!</p> <pre>AEGP_SetMemReportingOn(A_Boolean turn_OnB);</pre>

TABLE 84: AEGP_MEMORYSUITE1

Function	Purpose
AEGP_GetMemStats	Obtain information about the number of currently allocated handles and their total size. <pre>AEGP_GetMemStats(AEGP_MemID mem_id, A_long *countPL, A_long *sizePL);</pre>

MASK MANAGEMENT

Access, manipulate, and delete a layer's masks.

TABLE 85: AEGP_MASKSUITE5

Function	Purpose
AEGP_GetLayerNumMasks	Counts the masks applied to a layer, <pre>AEGP_GetLayerNumMasks(AEGP_LayerH aegp_layerH, A_long *num_masksPL);</pre>
AEGP_GetLayerMaskByIndex	Given a layer handle and mask index, returns a pointer to the mask handle. You must destroy the mask handle by using AEGP_DisposeMask() . <pre>AEGP_GetLayerMaskByIndex(AEGP_LayerH aegp_layerH, A_long mask_indexL, AEGP_MaskRefH *maskPH);</pre>
AEGP_DisposeMask	Dispose of a mask handle. <pre>AEGP_DisposeMask(AEGP_MaskRefH maskH);</pre>
AEGP_GetMaskInvert	Given a mask handle, determines if the mask is inverted or not. <pre>AEGP_GetMaskInvert(AEGP_MaskRefH maskH, A_Boolean *invertPB);</pre>
AEGP_SetMaskInvert	Sets the inversion state of a mask. <pre>AEGP_SetMaskInvert)(AEGP_MaskRefH mask_refH, A_Boolean invertB);</pre>

TABLE 85: AEGP_MASKSUITE5

Function	Purpose
AEGP_GetMaskMode	<p>Given a mask handle, returns the current mode of the mask. PF_MaskMode_NONE does nothing, PF_MaskMode_ADD is the default behavior.</p> <p>PF_MaskMode_NONE PF_MaskMode_ADD, PF_MaskMode_SUBTRACT, PF_MaskMode_INTERSECT, PF_MaskMode_LIGHTEN, PF_MaskMode_DARKEN, PF_MaskMode_DIFFERENCE,</p> <pre> AEGP_GetMaskMode(AEGP_MaskRefH maskH, PF_MaskMode *modeP); </pre>
AEGP_SetMaskMode	<p>Sets the mode of the given mask.</p> <pre> AEGP_SetMaskMode(AEGP_MaskRefH maskH, PF_MaskMode mode); </pre>
AEGP_GetMaskMotionBlurState	<p>Retrieves the motion blur setting for the given mask.</p> <pre> AEGP_GetMaskMotionBlurState(AEGP_MaskRefH mask_refH, AEGP_MaskMBlur *blur_stateP); </pre> <p>AEGP_MaskMBlur will be one of the following:</p> <p>AEGP_MaskMBlur_SAME_AS_LAYER AEGP_MaskMBlur_OFF AEGP_MaskMBlur_ON</p>
AEGP_SetMaskMotionBlurState	<p>Sets the motion blur setting for the given mask.</p> <pre> AEGP_SetMaskMotionBlurState(AEGP_MaskRefH mask_refH, AEGP_MaskMBlur blur_state); </pre>
AEGP_GetMaskName	<p>Removed in CS4. Use AEGP_GetNewStreamRefForMask and the name functions in the Dynamic Stream Suite instead.</p> <p>Given a mask handle, returns its name (up to AEGP_MAX_MASK_NAME_LEN + 1 long).</p> <pre> AEGP_GetMaskName(AEGP_MaskRefH mask_refH, A_char *nameZ); </pre>

TABLE 85: AEGP_MASKSUITE5

Function	Purpose
AEGP_SetMaskName	<p>Removed in CS4. Use AEGP_GetNewStreamRefForMask and the name functions in the Dynamic Stream Suite instead.</p> <p>Sets a given mask's name (up to AEGP_MAX_MASK_NAME_LEN + 1 long).</p> <pre>AEGP_GetMaskName(AEGP_MaskRefH mask_refH, A_char *nameZ);</pre>
AEGP_GetMaskID	<p>Retrieves the AEGP_MaskIDVal for the given AEGP_MaskRefH, for use in uniquely identifying the mask.</p> <pre>AEGP_GetMaskID(AEGP_MaskRefH mask_refH, AEGP_MaskIDVal *id_valP);</pre>
AEGP_CreateNewMask	<p>Creates a new mask on the referenced AEGP_LayerH, with zero nodes. The new mask's index is returned.</p> <pre>AEGP_CreateNewMask(AEGP_LayerH layerH, AEGP_MaskRefH *mask_refPH, A_long *mask_indexPL0);</pre>
AEGP_DeleteMaskFromLayer	<pre>AEGP_DeleteMaskFromLayer(AEGP_MaskRefH mask_refH);</pre> <p>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.</p>
AEGP_GetMaskColor	<p>Retrieves the color of the specified mask.</p> <pre>AEGP_GetMaskColor(AEGP_MaskRefH mask_refH, AEGP_ColorVal *colorP);</pre>
AEGP_SetMaskColor	<p>Sets the color of the specified mask.</p> <pre>AEGP_SetMaskColor(AEGP_MaskRefH mask_refH, const AEGP_ColorVal *colorP);</pre>
AEGP_GetMaskLockState	<p>Retrieves the lock state of the specified mask.</p> <pre>AEGP_GetMaskLockState(AEGP_MaskRefH mask_refH, A_Boolean *is_lockedPB);</pre>

TABLE 85: AEGP_MASKSUITE5

Function	Purpose
AEGP_SetMaskLockState	Sets the lock state of the specified mask. <pre>AEGP_SetMaskLockState(AEGP_MaskRefH mask_refH, A_Boolean lockB);</pre>
AEGP_GetMaskIsRotoBezier	Returns whether or not the given mask is used as a roto bezier. <pre>AEGP_GetMaskIsRotoBezier(AEGP_MaskRefH mask_refH, A_Boolean *is_roto_bezierPB);</pre>
AEGP_SetMaskIsRotoBezier	Sets whether a given mask is to be used as a roto bezier. <pre>AEGP_SetMaskIsRotoBezier(AEGP_MaskRefH mask_refH, A_Boolean *is_roto_bezierPB);</pre>
AEGP_DuplicateMask	Duplicates a given AEGP_MaskRefH. Caller must dispose of duplicate. <pre>AEGP_DuplicateMask(AEGP_MaskRefH orig_mask_refH, AEGP_MaskRefH *dupe_mask_refPH);</pre>

MASK OUTLINES

The Mask Suite 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 suite.

TABLE 86: AEGP_MASKOUTLINESUITE2

Function	Purpose
AEGP_IsMaskOutlineOpen	Given an mask outline pointer (obtainable through AEGP_StreamSuite), determines if the mask path is open or closed. <pre>AEGP_IsMaskOutlineOpen(AEGP_MaskOutlineVal *mask_outlineP, A_Boolean *openPB);</pre>
AEGP_SetMaskOutlineOpen	Sets the open state of the given mask outline. <pre>AEGP_SetMaskOutlineOpen(AEGP_MaskOutlineValH mask_outlineH, A_Boolean openB);</pre>

TABLE 86: AEGP_MASKOUTLINESUITE2

Function	Purpose
<code>AEGP_GetMaskOutlineNumSegments</code>	<p>Given a mask outline pointer, returns the number of segments in the path. <code>num_segmentsPL</code> is the total number of segments [0...N-1].</p> <pre>AEGP_GetMaskOutlineNumSegments(AEGP_MaskOutlineVal *mask_outlineP, A_long *num_segmentsPL);</pre>
<code>AEGP_GetMaskOutlineVertexInfo</code>	<p>Given a mask outline pointer and a point between 0 and the total number of segments. For closed mask paths, <code>vertex[0]</code> is the same as <code>vertex[num_segments]</code>.</p> <pre>AEGP_GetMaskOutlineVertexInfo(AEGP_MaskOutlineVal *mask_outlineP, A_long which_pointL, AEGP_MaskVertex *vertexP);</pre>
<code>AEGP_SetMaskOutlineVertexInfo</code>	<p>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, <code>which_pointL</code> must be valid for the mask outline, or the function will return an error.</p> <pre>AEGP_SetMaskOutlineVertexInfo(AEGP_MaskOutlineValH mask_outlineH, AEGP_VertexIndex which_pointL, AEGP_MaskVertex *vertexP);</pre>
<code>AEGP_CreateVertex</code>	<p>Creates a vertex at index position. All vertices which formerly had an <code>AEGP_VertexIndex</code> of position or greater will have their indices incremented by one.</p> <pre>AEGP_CreateVertex(AEGP_MaskOutlineValH mask_outlineH, AEGP_VertexIndex position);</pre> <p>NOTE: All masks must have at least one vertex.</p>
<code>AEGP_DeleteVertex</code>	<p>Removes a vertex from a mask.</p> <pre>AEGP_DeleteVertex(AEGP_MaskOutlineValH mask_outlineH, AEGP_VertexIndex index);</pre>

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_FpLong`s 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 87: AEGP_PERSISTENTDATASUITE3

Function	Purpose
<code>AEGP_GetApplicationBlob</code>	Obtains the handle to all persistent application data. <code>AEGP_GetApplicationBlob(AEGP_PersistentBlobH *blobPH);</code>
<code>AEGP_GetNumSections</code>	Obtains the number of sections in the application blob. <code>AEGP_GetNumSections(AEGP_PersistentBlobH blobH, A_long *num_sectionPL);</code>
<code>AEGP_GetSectionKeyByIndex</code>	Obtains the key at the given index. <code>AEGP_GetSectionKeyByIndex(AEGP_PersistentBlobH blobH, A_long section_index, A_long max_section_size, A_char *section_keyZ);</code>
<code>AEGP_DoesKeyExist</code>	Returns whether or not a given key/value pair exists with the blob. <code>AEGP_DoesKeyExist(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, A_Boolean *existsPB);</code>

TABLE 87: AEGP_PERSISTENTDATA3

Function	Purpose
AEGP_GetNumKeys	Retrieves the number of value keys in the section. <pre>AEGP_GetNumKeys(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, A_long *num_keysPL);</pre>
AEGP_GetValueKeyByIndex	Retrieves the value of the indexed key. <pre>AEGP_GetValueKeyByIndex(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, A_long key_index, A_long max_key_size, A_char *value_keyZ);</pre>
<i>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.</i>	
AEGP_GetDataHandle	Obtains the value associated with the given section's key. If using in-memory data structures, watch for endian issues. <pre>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);</pre>
AEGP_GetData	Obtains the data located at a given section's value. <pre>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);</pre>
AEGP_GetString	Obtains the string for a given section key's value (and indicates its length in actual_szLu0). <pre>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);</pre>

TABLE 87: AEGP_PERSISTENTDATA SUITE3

Function	Purpose
AEGP_GetLong	Obtains the A_long associated with a given section key's value. <pre> AEGP_GetLong(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, A_long defaultL, A_long *valuePL); </pre>
AEGP_GetFpLong	Obtains the A_FpLong associated with a given section key's value. <pre> AEGP_GetFpLong(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, A_FpLong defaultF, A_FpLong *valuePF); </pre>
AEGP_SetDataHandle	Sets the given section key's value to the handle passed in. <pre> AEGP_SetDataHandle(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, const AEGP_MemHandle valueH); </pre>
AEGP_SetData	Sets the given section key's value to the data contained in dataPV. <pre> AEGP_SetData(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, A_u_long data_sizeLu, const void *dataPV); </pre>
AEGP_SetString	Sets the given section key's string to strZ. <pre> AEGP_SetString(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, const A_char *strZ); </pre>

TABLE 87: AEGP_PERSISTENTDATA SUITE3

Function	Purpose
<code>AEGP_SetLong</code>	Sets the given section key's value to valueL. <pre>AEGP_SetLong(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, A_long valueL);</pre>
<code>AEGP_SetFpLong</code>	Sets the given section key's value to valueF. <pre>AEGP_SetFpLong(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ, A_FpLong valueF);</pre>
<code>AEGP_DeleteEntry</code>	Removes the given section's value from the blob. <pre>AEGP_DeleteEntry(AEGP_PersistentBlobH blobH, const A_char *section_keyZ, const A_char *value_keyZ);</pre>
<code>AEGP_GetPrefsDirectory</code>	New in CS3 (8.0). 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 <code>AEGP_FreeMemHandle</code> . <pre>AEGP_GetPrefsDirectory)(AEGP_MemHandle *unicode_pathPH);</pre>

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 88: AEGP_PFINTERFACESUITE1

Function	Purpose
<code>AEGP_GetEffectLayer</code>	Obtain the layer handle of the layer to which the effect is applied. <pre>AEGP_GetEffectLayer(PF_ProgPtr effect_ref, AEGP_LayerH *layerPH);</pre>
<code>AEGP_GetNewEffectForEffect</code>	Obtain the <code>AEGP_EffectRefH</code> corresponding to the effect. <pre>AEGP_GetNewEffectForEffect(AEGP_PluginID aegp_plugin_id, PF_ProgPtr effect_ref, AEGP_EffectRefH *effectPH);</pre>
<code>AEGP_ConvertEffectToCompTime</code>	Retrieve the composition time corresponding to the effect's layer time. <pre>AEGP_ConvertEffectToCompTime(PF_ProgPtr effect_ref, long what_timeL, unsigned long time_scaleLu, A_Time *comp_timePT);</pre>
<code>AEGP_GetEffectCamera</code>	Obtain the camera (if any) being used by After Effects to view the effect's layer. <pre>AEGP_GetEffectCamera(PF_ProgPtr effect_ref, const A_Time *comp_timePT, AEGP_LayerH camera_layerPH);</pre>
<code>AEGP_GetEffectCameraMatrix</code>	Obtain the transform used to move between the layer's coordinate space and that of the containing composition. <pre>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);</pre> <p>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).</p>

AEGP_GETEFFECTCAMERAMATRIX NOTES

The model view for the camera matrix is inverse of the matrix obtained from [AEGP_GetEffectCameraMatrix\(\)](#). Also note that our matrix is row-based; OpenGL's is column-based.

RENDER QUEUE SUITE

This suite allows AEGPs to add items to the render queue (using default options), and control the state of the render queue.

TABLE 89: AEGP_RENDERQUEUESUITE1

Function	Purpose
AEGP_AddCompToRenderQueue	Adds a composition to the render queue, using default options. AEGP_AddCompToRenderQueue(AEGP_CompHcompH, 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_RenderQueueStatestate); AEGP_RenderQueueState_STOPPED AEGP_RenderQueueState_PAUSED AEGP_RenderQueueState_RENDERING
AEGP_GetRenderQueueState	Obtains the current render queue state. AEGP_SetRenderQueueState(AEGP_RenderQueueStatestate);

RENDER QUEUE ITEM SUITE

Manipulate all aspects of render queue items using this suite.

TABLE 90: AEGP_RQITEMSUITE3

Function	Purpose
AEGP_GetNumRQItems	Returns the number of items currently in the render queue. <pre>AEGP_GetNumRQItems (A_long *num_itemsPL);</pre>
AEGP_GetRQItemByIndex	Returns an AEGP_RQItemRefH referencing the index'd item. <pre>AEGP_GetRQItemByIndex (A_long rq_item_index, AEGP_RQItemRefH *rq_item_refPH);</pre>
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. <pre>AEGP_GetNextRQItem (AEGP_RQItemRefH current_rq_itemH, AEGP_RQItemRefH *next_rq_itemPH);</pre>
AEGP_GetNumOutputModulesForRQItem	Returns the number of output modules applied to the given AEGP_RQItemRefH. <pre>AEGP_GetNumOutputModulesForRQItem (AEGP_RQItemRefH rq_itemH, A_long *num_outmodsPL);</pre>
AEGP_GetRenderState	Returns TRUE if the AEGP_RQItemRefH is set to render (once the user clicks the Render button). <pre>AEGP_GetRenderState (AEGP_RQItemRefH rq_itemH, A_Boolean *will_renderPB);</pre>
AEGP_SetRenderState	New in 7.0. Controls whether or not the AEGP_RQItemRefH will render when the user next clicks the Render button. Returns an error if called during rendering. As of 7.0 (and AEGP_RQItemSuite3), 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) <pre>AEGP_SetRenderState (AEGP_RQItemRefH rq_itemH, A_Boolean renderB);</pre>

TABLE 90: AEGP_RQITEMSUITE3

Function	Purpose
AEGP_GetStartedTime	Returns the time (in seconds, since 1904) that rendering began. <pre>AEGP_GetStartedTime(AEGP_RQItemRefH rq_itemH, A_Time *started_timePT);</pre>
AEGP_GetElapsedTime	Returns the time elapsed since rendering began. <pre>AEGP_GetElapsedTime(AEGP_RQItemRefH rq_itemH, A_Time *render_timePT);</pre>
AEGP_GetLogType	Returns the log type for the referenced AEGP_RQItemRefH. <pre>AEGP_GetLogType(AEGP_RQItemRefH rq_itemH, AEGP_LogType *logtypeP);</pre> <p>AEGP_LogType will have one of the following values:</p> <p>AEGP_LogType_NONE AEGP_LogType_ERRORS_ONLY AEGP_LogType_PLUS_SETTINGS AEGP_LogType_PER_FRAME_INFO</p>
AEGP_SetLogType	Specifies the log type to be used with the referenced AEGP_RQItemRefH. <pre>AEGP_SetLogType(AEGP_RQItemRefH rq_itemH, AEGP_LogType logtype);</pre>
AEGP_RemoveOutputModule	Removes the specified output module from the referenced AEGP_RQItemRefH. <pre>AEGP_RemoveOutputModule(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH);</pre>
AEGP_GetComment	Retrieves the comment associated with the referenced AEGP_RQItemRefH. <pre>AEGP_GetComment(AEGP_RQItemRefH rq_itemH, A_char *commentZ);</pre>
AEGP_SetComment	Specifies the comment associated with the referenced AEGP_RQItemRefH. <pre>AEGP_SetComment(AEGP_RQItemRefH rq_itemH, const A_char *commentZ);</pre>

TABLE 90: AEGP_RQITEMSUITE3

Function	Purpose
AEGP_GetCompFromRQItem	Retrieves the AEGP_CompH associated with the AEGP_RQItemRefH. <pre>AEGP_GetCompFromRQItem(AEGP_RQItemRefH rq_itemH, AEGP_CompH *compPH);</pre>
AEGP_DeleteRQItem	New in 7.0. Deletes the render queue item. Undoable. <pre>AEGP_DeleteRQItem(AEGP_RQItemRefH rq_itemH);</pre>

RENDER OPTIONS SUITE

For each render queue item, there is a set of associated options. The Render Options Suite allows you to get and set those options.

TABLE 91: AEGP_RENDEROPTIONSUIE3

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. <pre>AEGP_NewFromItem(AEGP_PluginID plugin_id, AEGP_ItemH itemH, AEGP_RenderOptionsH *optionsPH);</pre>
AEGP_Duplicate	Duplicates an AEGP_RenderOptionsH into copyPH. <pre>AEGP_Duplicate(AEGP_PluginID plugin_id, AEGP_RenderOptionsH optionsH, AEGP_RenderOptionsH *copyPH);</pre>
AEGP_Dispose	Deletes an AEGP_RenderOptionsH. <pre>AEGP_Dispose(AEGP_RenderOptionsH optionsH);</pre>

TABLE 91: AEGP_RENDEROPTIONSUI3

Function	Purpose
AEGP_SetTime	Sets the render time of an AEGP_RenderOptionsH. <pre>AEGP_SetTime(AEGP_RenderOptionsH optionsH, A_Time time);</pre>
AEGP_GetTime	Retrieves the render time of the given AEGP_RenderOptionsH. <pre>AEGP_GetTime(AEGP_RenderOptionsH optionsH, A_Time *timeP);</pre>
AEGP_SetTimeStep	Specifies the time step (duration of a frame) for the referenced AEGP_RenderOptionsH. <pre>AEGP_SetTimeStep(AEGP_RenderOptionsH optionsH, A_Time time_step);</pre>
AEGP_GetTimeStep	Retrieves the time step (duration of a frame) for the given AEGP_RenderOptionsH. <pre>AEGP_GetTimeStep(AEGP_RenderOptionsH optionsH, A_Time *timePT);</pre>
AEGP_SetFieldRender	Specifies the field settings for the given AEGP_RenderOptionsH. <pre>AEGP_SetFieldRender(AEGP_RenderOptionsH optionsH, PF_Field field_render);</pre>
AEGP_GetFieldRender	Retrieves the field settings for the given AEGP_RenderOptionsH. <pre>AEGP_GetFieldRender(AEGP_RenderOptionsH optionsH, PF_Field *field_renderP);</pre>
AEGP_SetWorldType	Specifies the AEGP_WorldType of the output of a given AEGP_RenderOptionsH. <pre>AEGP_SetWorldType(AEGP_RenderOptionsH optionsH, AEGP_WorldType type);</pre> <p>AEGP_WorldType will be either AEGP_WorldType_8 or AEGP_WorldType_16</p>
AEGP_GetWorldType	Retrieves the AEGP_WorldType of the given AEGP_RenderOptionsH. <pre>AEGP_GetWorldType(AEGP_RenderOptionsH optionsH, AEGP_WorldType *typeP);</pre>

TABLE 91: AEGP_RENDEROPTIONS SUITE 3

Function	Purpose
AEGP_SetDownsampleFactor	<p>Specifies the downsample factor (with independent horizontal and vertical settings) for the given AEGP_RenderOptionsH.</p> <pre> AEGP_SetDownsampleFactor(AEGP_RenderOptionsH optionsH, A_short x, A_short y); </pre>
AEGP_GetDownsampleFactor	<p>Retrieves the downsample factor for the given AEGP_RenderOptionsH.</p> <pre> AEGP_GetDownsampleFactor(AEGP_RenderOptionsH optionsH, A_short *xP, A_short *yP); </pre>
AEGP_SetRegionOfInterest	<p>Specifies the region of interest sub-rectangle for the given AEGP_RenderOptionsH.</p> <pre> AEGP_SetRegionOfInterest(AEGP_RenderOptionsH optionsH, const A_LRect *roiP) </pre>
AEGP_GetRegionOfInterest	<p>Retrieves the region of interest sub-rectangle for the given AEGP_RenderOptionsH.</p> <pre> AEGP_GetRegionOfInterest(AEGP_RenderOptionsH optionsH, A_LRect *roiP); </pre>
AEGP_SetMatteMode	<p>Specifies the AEGP_MatteMode for the given AEGP_RenderOptionsH.</p> <pre> AEGP_SetMatteMode(AEGP_RenderOptionsH optionsH, AEGP_MatteMode mode); </pre> <p>AEGP_MatteMode will be one of the following:</p> <pre> AEGP_MatteMode_STRAIGHT AEGP_MatteMode_PREMUL_BLACK AEGP_MatteMode_PREMUL_BG_COLOR </pre>
AEGP_GetMatteMode	<p>Retrieves the AEGP_MatteMode for the given AEGP_RenderOptionsH.</p> <pre> AEGP_GetMatteMode(AEGP_RenderOptionsH optionsH, AEGP_MatteMode *modeP); </pre>

TABLE 91: AEGP_RENDEROPTIONSUI3

Function	Purpose
AEGP_GetChannelOrder	Gets the AEGP_ChannelOrder for the given AEGP_RenderOptionsH. AEGP_ChannelOrder will be either AEGP_ChannelOrder_ARGB or AEGP_ChannelOrder_BGRA. <pre>AEGP_GetChannelOrder(AEGP_RenderOptionsH optionsH, AEGP_ChannelOrder *orderP);</pre> <p>Factoid: this was added to facilitate live linking with Premiere Pro.</p>
AEGP_SetChannelOrder	Sets the AEGP_ChannelOrder of the AEGP_RenderOptionsH. <pre>AEGP_SetChannelOrder(AEGP_RenderOptionsH optionsH, AEGP_ChannelOrder order);</pre>
AEGP_GetRenderGuideLayers	Passes back a boolean that is true if the render guide layers setting is on. <pre>AEGP_GetRenderGuideLayers)(AEGP_RenderOptionsH optionsH, A_Boolean *will_renderPB);</pre>
AEGP_SetRenderGuideLayers	Specify whether or not to render guide layers. <pre>AEGP_SetRenderGuideLayers)(AEGP_RenderOptionsH optionsH, A_Boolean render_themB);</pre>

OUTPUT MODULE SUITE

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 92: AEGP_OUTPUTMODULESUITE4

Function	Purpose
AEGP_GetOutputModuleByIndex	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. <pre>AEGP_GetOutputModuleByIndex(AEGP_RQItemRefH rq_itemH, A_long outmod_indexL, AEGP_OutputModuleRefH *outmodPH);</pre>
AEGP_GetEmbedOptions	Retrieves the embedding setting specified for the referenced AEGP_OutputModuleRefH. <pre>AEGP_GetEmbedOptions(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_EmbeddingType *embed_optionsP);</pre> <p>AEGP_EmbeddingType will be one of the following:</p> <p>AEGP_Embedding_NOTHING AEGP_Embedding_LINK AEGP_Embedding_LINK_AND_COPY</p>
AEGP_SetEmbedOptions	Specifies the embedding setting for the referenced AEGP_OutputModuleRefH. <pre>AEGP_SetEmbedOptions(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_EmbeddingType embed_options);</pre>
AEGP_GetPostRenderAction	Retrieves the post-render action setting for the referenced AEGP_OutputModuleRefH. <pre>AEGP_GetPostRenderAction(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_PostRenderAction *actionP);</pre> <p>AEGP_PostRenderAction will be one of the following:</p> <p>AEGP_PostRenderOptions_IMPORT AEGP_PostRenderOptions_IMPORT_AND_REPLACE_USAGE AEGP_PostRenderOptions_SET_PROXY</p>

TABLE 92: AEGP_OUTPUTMODULESUITE4

Function	Purpose
<code>AEGP_SetPostRenderAction</code>	<p>Specifies the post-render action setting for the referenced <code>AEGP_OutputModuleRefH</code>.</p> <pre>AEGP_SetPostRenderAction(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_PostRenderAction action);</pre>
<code>AEGP_GetEnabledOutputs</code>	<p>Retrieves which output types are enabled for the referenced <code>AEGP_OutputModuleRefH</code>.</p> <pre>AEGP_GetEnabledOutputs(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_OutputTypes *typesP);</pre> <p><code>AEGP_OutputTypes</code> will contain one or both of the following values:</p> <p><code>AEGP_OutputType_VIDEO</code> <code>AEGP_OutputType_AUDIO</code></p> <p>NOTE: These are flags, not an enumeration.</p>
<code>AEGP_SetEnabledOutputs</code>	<p>Specifies which output types are enabled for the referenced <code>AEGP_OutputModuleRefH</code>.</p> <pre>AEGP_SetEnabledOutputs(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_OutputTypes enabled_types);</pre>
<code>AEGP_GetOutputChannels</code>	<p>Retrieves which video channels are enabled for output in the referenced <code>AEGP_OutputModuleRefH</code>.</p> <pre>AEGP_GetOutputChannels(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_VideoChannels *outchannelsP);</pre> <p><code>AEGP_VideoChannels</code> will be one of the following:</p> <p><code>AEGP_VideoChannels_RGB</code> <code>AEGP_VideoChannels_RGBA</code> <code>AEGP_VideoChannels_ALPHA</code></p>
<code>AEGP_SetOutputChannels</code>	<p>Specifies which video channels are enabled for output in the referenced <code>AEGP_OutputModuleRefH</code>.</p> <pre>AEGP_SetOutputChannels(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_VideoChannels outchannels);</pre>

TABLE 92: AEGP_OUTPUTMODULESUITE4

Function	Purpose
AEGP_GetStretchInfo	<p>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.</p> <pre> AEGP_GetStretchInfo(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, A_Boolean *enabledPB, AEGP_StretchQuality *qualP, A_Boolean *lockedPB); </pre> <p>AEGP_StretchQuality will be one of the following: AEGP_StretchQual_LOW AEGP_StretchQual_HIGH</p>
AEGP_SetStretchInfo	<p>Retrieves the stretch information enabled for the referenced AEGP_OutputModuleRefH.</p> <pre> AEGP_SetStretchInfo(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, A_Boolean is_enabledB, AEGP_StretchQuality quality); </pre>
AEGP_GetCropInfo	<p>Retrieves whether or not the cropping is enabled for the referenced AEGP_OutputModuleRefH, and the rectangle to be used.</p> <pre> AEGP_GetCropInfo(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, A_Boolean *is_enabledBP, A_Rect *crop_rectP); </pre>
AEGP_SetCropInfo	<p>Specifies whether cropping is enabled for the referenced AEGP_OutputModuleRefH, and the rectangle to be used.</p> <pre> AEGP_SetCropInfo(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, A_Boolean enableB, A_Rect crop_rect); </pre>
AEGP_GetSoundFormatInfo	<p>Retrieves whether or not audio output is enabled for the referenced AEGP_OutputModuleRefH, and the settings to be used.</p> <pre> AEGP_GetSoundFormatInfo(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_SoundDataFormat *formatP, A_Boolean *enabledPB); </pre>

TABLE 92: AEGP_OUTPUTMODULESUITE4

Function	Purpose
<code>AEGP_SetSoundFormatInfo</code>	<p>Specifies whether or not audio output is enabled for the referenced <code>AEGP_OutputModuleRefH</code>, and the settings to be used.</p> <pre> AEGP_SetSoundFormatInfo(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_SoundDataFormat format_info, A_Boolean enabledB); </pre>
<code>AEGP_GetOutputFilePath</code>	<p>Retrieves the path to which <code>AEGP_OutputModuleRefH</code>'s output file will be written. The path is a handle to a NULL-terminated <code>A_UTF16Char</code> string, and must be disposed with <code>AEGP_FreeMemHandle</code>.</p> <pre> AEGP_GetOutputFilePath(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, AEGP_MemHandle *unicode_pathPH); </pre>
<code>AEGP_SetOutputFilePath</code>	<p>Specifies the path to which <code>AEGP_OutputModuleRefH</code>'s output file will be written. The file path is a NULL-terminated UTF-16 string with platform separators.</p> <pre> AEGP_SetOutputFilePath(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH outmodH, const A_UTF16Char *pathZ); </pre>
<code>AEGP_AddDefaultOutputModule</code>	<p>Adds the default output module to the specified <code>AEGP_RQItemRefH</code>, and returns the added output module's <code>AEGP_OutputModuleRefH</code> (you wouldn't add it if you didn't plan to mess around with it, would you?).</p> <pre> AEGP_AddDefaultOutputModule(AEGP_RQItemRefH rq_itemH, AEGP_OutputModuleRefH *outmodPH); </pre>
<code>AEGP_GetExtraOutputModuleInfo</code>	<p>Retrieves information about the output module. <code>format_uniPH</code> and <code>info_uniPH</code> provide the textual description of, and information about, the output module, formatted as the user would see it. <code>format_uniPH</code> and <code>info_uniPH</code> will contain NULL-terminated UTF16 strings, of which the caller must dispose.</p> <pre> 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); </pre>

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 93: AEGP_COLLECTIONSUITE2

Function	Purpose
<code>AEGP_NewCollection</code>	Creates and returns a new, empty collection. To obtain the current composition's selection as a collection, use AEGP_GetNewCollectionFromCompSelection . <pre>AEGP_NewCollection(AEGP_PluginID plugin_id, AEGP_Collection2H *collectionPH);</pre>
<code>AEGP_DisposeCollection</code>	Disposes of a collection. <pre>AEGP_DisposeCollection(AEGP_Collection2H collectionH);</pre>
<code>AEGP_GetCollectionNumItems</code>	Returns the number of items contained in the given collection. <pre>AEGP_GetCollectionNumItems(AEGP_Collection2H collectionH, A_u_long *num_itemsPL);</pre>
<code>AEGP_GetCollectionItemByIndex</code>	Retrieves (creates and populates) the index'd collection item. <pre>AEGP_GetCollectionItemByIndex(AEGP_Collection2H collectionH, A_u_long indexL, AEGP_CollectionItemV2 *itemP);</pre>

TABLE 93: AEGP_COLLECTIONSUITE2

Function	Purpose
<code>AEGP_CollectionPushBack</code>	<p>Adds an item to the given collection.</p> <pre>AEGP_CollectionPushBack(AEGP_Collection2H collectionH, const AEGP_CollectionItemV2 *itemP);</pre>
<code>AEGP_CollectionErase</code>	<p>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.</p> <pre>AEGP_CollectionErase(AEGP_Collection2H collectionH, A_u_long index_firstL, A_u_long index_lastL);</pre>

OWNERSHIP OF COLLECTION ITEMS

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`.

THE AEGP_WORLD AS WE KNOW IT

`AEGP_Worlds` are the common format used throughout the `AEGP` APIs to describe frames of pixels.

TABLE 94: AEGP_WORLD SUITE3

Function	Purpose
<code>AEGP_New</code>	<p>Returns an allocated, initialized <code>AEGP_WorldH</code>.</p> <pre>AEGP_New(AEGP_PluginID plugin_id, AEGP_WorldType type, A_long widthL, A_long heightL, AEGP_WorldH *worldPH);</pre>
<code>AEGP_Dispose</code>	<p>Disposes of an <code>AEGP_WorldH</code>. Use this on every world you allocate.</p> <pre>AEGP_Dispose(AEGP_WorldH worldH);</pre>

TABLE 94: AEGP_WORLD SUITE3

Function	Purpose
AEGP_GetType	<p>Returns the type of a given AEGP_WorldH.</p> <pre> AEGP_GetType(AEGP_WorldH worldH, AEGP_WorldType **typeP); </pre> <p>AEGP_WorldType will be one of the following:</p> <pre> AEGP_WorldType_8, AEGP_WorldType_16, AEGP_WorldType_32 </pre>
AEGP_GetSize	<p>Returns the width and height of the given AEGP_WorldH.</p> <pre> AEGP_GetSize(AEGP_WorldH worldH, A_long *widthPL, A_long *heightPL); </pre>
AEGP_GetRowBytes	<p>Returns the rowbytes for the given AEGP_WorldH.</p> <pre> AEGP_GetRowBytes(AEGP_WorldH worldH, A_u_long *row_bytesPL); </pre>
AEGP_GetBaseAddr8	<p>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.</p> <pre> AEGP_GetBaseAddr8(AEGP_WorldH worldH, PF_Pixel8 **base_addrP); </pre>
AEGP_GetBaseAddr16	<p>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.</p> <pre> AEGP_GetBaseAddr16(AEGP_WorldH worldH, PF_Pixel16 **base_addrP); </pre>
AEGP_GetBaseAddr32	<p>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.</p> <pre> AEGP_GetBaseAddr32(AEGP_WorldH worldH, PF_PixelFloat **base_addrP); </pre>

TABLE 94: AEGP_WORLD SUITE 3

Function	Purpose
<code>AEGP_FillOutPFEffectWorld</code>	<p>Populates and returns a <code>PF_EffectWorld</code> representing the given <code>AEGP_WorldH</code>, for use with numerous pixel processing callbacks.</p> <p>NOTE: This does not give your plug-in ownership of the world referenced; destroy the source <code>AEGP_WorldH</code> only if you allocated it. It just fills out the provided <code>PF_EffectWorld</code> to point to the same pixel buffer.</p> <pre>AEGP_FillOutPFEffectWorld(AEGP_WorldH worldH, PF_EffectWorld *pf_worldP);</pre>
<code>AEGP_FastBlur</code>	<p>Performs a fast blur on a given <code>AEGP_WorldH</code>.</p> <pre>AEGP_FastBlur(A_FpLong radiusF, PF_ModeFlags mode, PF_Quality quality, AEGP_WorldH worldH);</pre>
<code>AEGP_NewPlatformWorld</code>	<p>Creates a new <code>AEGP_PlatformWorldH</code> (a pixel world native to the execution platform).</p> <pre>AEGP_NewPlatformWorld(AEGP_PluginID plugin_id, AEGP_WorldType type, A_long widthL, A_long heightL, AEGP_PlatformWorldH *worldPH);</pre>
<code>AEGP_DisposePlatformWorld</code>	<p>Disposes of an <code>AEGP_PlatformWorldH</code>.</p> <pre>AEGP_DisposePlatformWorld(AEGP_PlatformWorldH worldH);</pre>
<code>AEGP_NewReferenceFromPlatformWorld</code>	<p>Retrieves an <code>AEGP_WorldH</code> referring to the given <code>AEGP_PlatformWorldH</code>. NOTE: This doesn't allocate a new world, it simply provides a reference to an existing one.</p> <pre>AEGP_NewReferenceFromPlatformWorld(AEGP_PluginID plugin_id, AEGP_PlatformWorldH plat_worldH, AEGP_WorldH *worldPH);</pre>

DO THIS MANY TIMES

Utilizes multiple processors (if available) for your computations.

TABLE 95: AEGP_ITERATESUITE1

Function	Purpose
AEGP_GetNumThreads	Ask After Effects how many threads are currently available. <pre>AEGP_GetNumThreads(A_long *num_threadsPL);</pre>
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 for a description of how refconPV is used. <pre>AEGP_IterateGeneric(A_longssg iterationsL, void *refconPV, A_Err (*fn_func) (void *refconPV, A_long thread_indexL, A_long i, A_long iterationsL));</pre>

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](#) and [AEGP_GetCompBGColor](#) functions. This should not be interpreted to mean that effects can use *any* AEGP suite calls. Also, see the [Events chapter](#) for more information on effects adding keyframes.

[AEGP_PFInterfaceSuite](#) 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](#) 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\(\)](#) for an example.

MODIFYING ITEMS IN THE RENDER QUEUE

If you call [AEGP_AddCompToRenderQueue](#) (from [AEGP_RenderQueueSuite](#)), 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](#). Consequently, their names cannot be retrieved by [AEGP_GetItemName](#) or [AEGP_GetLayerName](#). However, you can use the `ItemH` associated with them to [AEGP_GetItemName](#).

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\(\)](#) and [AEGP_SetLayerInPointAndDuration\(\)](#) to properly set the layer's time information.

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.

TRACKED MEMORY

Use the [memory suite](#) to allocate and free memory. You'll receive the benefits of our memory leak testing, debugging is simpler, and the blame for any memory errors you may cause.

ACCESSING PIXELS FROM EFFECT LAYER PARAMETERS

This has gotten decidedly easier in 7.0. Use [AEGP_GetNewStreamValue](#) to get the layer's `layer_id`, then the new [AEGP_GetLayerFromLayerID](#) to get the `AEGP_LayerH`.

8 • ARTISANS

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).

NOTE: This API is a little treacherous, and so far has only been used on one in-house project. If you're considering writing an Interactive Artisan, talk it over with us first.

WHAT'S NEW IN 7.0?

The ability to get and set channel order has been added to [AEGP_RenderOptionsSuite](#).

ARTISAN DATA TYPES

Below are the data types most commonly used in the Artisan API.

TABLE 96: 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.
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\(\)](#).

This tables shows the functions that Artisans can support as defined by `PR_ArtisanEntryPoints`: only [render_func](#) is required.

TABLE 97: ARTISAN ENTRY POINTS

PR_ArtisanEntryPoints	
<code>global_setup_func0</code>	<p>Called only once, right after <code>GP_Main</code>. 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 <code>global_setdown_func</code>.</p> <pre>PR_GlobalSetupFunc(const PR_InData *in_dataP, PR_GlobalContextH global_contextH, PR_GlobalDataH *global_dataPH);</pre>
<code>global_setdown_func0</code>	<p>Dispose of any global data you allocated.</p> <pre>PR_GlobalSetdownFunc(const PR_InData *in_dataP, PR_GlobalContextH global_contextH, PR_GlobalDataH global_dataH);</pre>
<code>global_do_about_func0</code>	<p>Tell the world about yourself! Use <code>in_dataP>msg_func</code> to display your dialog.</p> <pre>PR_GlobalDoAboutFunc(const PR_InData *in_dataP, PR_GlobalContextH global_contextH, PR_GlobalDataH global_dataH);</pre>
<code>setup_instance_func0</code>	<p>Allocate and instantiate any data specific to this instance of your Artisan.</p> <pre>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);</pre>

TABLE 97: ARTISAN ENTRY POINTS

PR_ArtisanEntryPoints	
setdown_instance_func0	<p>Deallocate and free any data specific to this instance of your Artisan.</p> <pre>PR_InstanceSetdownFunc(const PR_InData *in_dataP, PR_GlobalContextH global_contextH, PR_InstanceContextH instance_contextH, PR_GlobalDataH global_dataH, PR_InstanceDataH instance_dataH);</pre>
flatten_instance_func0	<p>Flatten your data in preparation to being written to disk. (making sure it's OS independent, if your Artisan is).</p> <pre>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);</pre>
do_instance_dialog_func0	<p>If your Artisan has a additional parameters (accessed through its Options dialog), this function will be called to get and set them.</p> <pre>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);</pre> <p>PR_DialogResult is either PR_DialogResult_NO_CHANGE or PR_DialogResult_CHANGE_MADE.</p>
frame_setup_func0	<p>Perform any setup necessary to render a frame (called immediately before rendering).</p> <pre>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);</pre>

TABLE 97: ARTISAN ENTRY POINTS

PR_ArtisanEntryPoints	
frame_setdown_func0	<p>Dispose of any setup data allocated during frame_setup (sent immediately after rendering).</p> <pre>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);</pre>
render_func	<p>Render the scene.</p> <pre>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);</pre>
query_func0	<p>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).</p> <pre>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);</pre> <p>PR_QueryType can be one of the following:</p> <pre>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</pre>

RENDER SUITE

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.

Since we introduced the AEGP API, we've been asked to provide functions for retrieving rendered frames. This function suite allows you to do just that.

TABLE 98: AEGP_RENDERSUITE

Function	Purpose
<code>AEGP_RenderAndCheckoutFrame</code>	<p>Retrieves an <code>AEGP_FrameReceiptH</code> (<i>not</i> the actual pixels) for the frame requested. Optionally, the AEGP can pass a function to be called by After Effects if the user cancels the current render, as well as a <code>refcon</code> (constant reference to opaque data) for use during that function.</p> <pre>AEGP_RenderAndCheckoutFrame(AEGP_RenderOptionsH optionsH, AEGP_RenderSuiteCheckForCancel cancel_funcP0, AEGP_CancelRefcon refconP0, AEGP_FrameReceiptH *receiptPH);</pre>
<code>AEGP_CheckinFrame</code>	<p>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!</p> <pre>AEGP_CheckinFrame(AEGP_FrameReceiptH receiptH);</pre>
<code>AEGP_GetReceiptWorld</code>	<p>Retrieves the pixels (<code>AEGP_WorldH</code>) associated with the referenced <code>AEGP_FrameReceiptH</code>.</p> <pre>AEGP_GetReceiptWorld(AEGP_FrameReceiptH receiptH, AEGP_WorldH *worldPH);</pre>
<code>AEGP_GetRenderedRegion</code>	<p>Retrieves an <code>A_LRect</code> containing the region of the <code>AEGP_FrameReceiptH</code>'s <code>AEGP_WorldH</code> that has already been rendered. Remember that it is increasingly 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.</p> <pre>AEGP_GetRenderedRegion(AEGP_FrameReceiptH receiptH, A_LRect *rendered_regionP);</pre>

TABLE 98: AEGP_RENDERSUITE

Function	Purpose
<code>AEGP_IsRenderedFrameSufficient</code>	<p>Given two sets of <code>AEGP_RenderOptionsH</code>, After Effects will return TRUE if the already-rendered pixels are still valid for the proposed <code>AEGP_RenderOptionsH</code>.</p> <pre> AEGP_IsRenderedFrameSufficient(AEGP_RenderOptionsH rendered_optionsH, AEGP_RenderOptionsH proposed_optionsH, A_Boolean *sufficientPB); </pre>
<code>AEGP_RenderNewItemSoundData</code>	<p>Obtains an <code>ItemH</code>'s audio at the given time, of the given duration, in the given format. The plug-in must dispose of the returned <code>AEGP_SoundDataH</code> (which may be NULL if no audio is available).</p> <pre> AEGP_RenderNewItemSoundData(AEGP_ItemH itemH, const A_Time *start_timePT, / const A_Time *durationPT, const AEGP_SoundDataFormat *formatP, AEGP_SoundDataH *new_dataPH); </pre>

THE WORLD IS YOUR CANVAS

[`AEGP_RenderTexture\(\)`](#) 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.

The following suite supplies the layers, compositions, texture and destination buffers. This is a vital suite for all artisans.

TABLE 99: AEGP_CANVASSUITE7

Function	Purpose
<code>AEGP_GetCompToRender</code>	<p>Given the render context provided to the Artisan at render time, returns a handle to the composition.</p> <pre> AEGP_GetCompToRender(PR_RenderContextH render_contextH, AEGP_CompH *compPH) </pre>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
<code>AEGP_GetNumLayersToRender</code>	<p>Given the render context, returns the number of layers the Artisan needs to render.</p> <pre> AEGP_GetNumLayersToRender (PR_RenderContextH render_contextH, A_long *num_to_renderPL) </pre>
<code>AEGP_GetNthLayerContextToRender</code>	<p>Used to build a list of layers to render after determining the total number of layers that need rendering by the Artisan.</p> <pre> AEGP_GetNthLayerContextToRender (PR_RenderContextH render_contextH, A_long n, A_long *layer_indexPL) </pre>
<code>AEGP_GetLayerFromLayerContext</code>	<p>Given a <code>AEGP_RenderLayerContextH</code>, retrieves the associated <code>AEGP_LayerH</code> (required by many suite functions).</p> <pre> AEGP_GetLayerFromLayerContext (const PR_RenderContextH render_contextH, AEGP_RenderLayerContextH layer_contextH, AEGP_LayerH *layerPH) ; </pre>
<code>AEGP_GetLayerAndSubLayerFromLayerContext</code>	<p>Allows for rendering of sub-layers (as within a Photoshop file).</p> <pre> AEGP_GetLayerAndSubLayerFromLayerContext (const PR_RenderContextH render_contextH, AEGP_RenderLayerContextH layer_contextH, AEGP_LayerH *layerPH, AEGP_SubLayerIndex *sublayerP) ; </pre>
<code>AEGP_GetTopLayerFromLayerContext</code>	<p>With collapsed geometrics “on” this gives the layer in the root composition containing the layer context. With collapsed geometrics off this is the same as AEGP_GetLayerFromLayerContext.</p> <pre> AEGP_GetTopLayerFromLayerContext (const PR_RenderContextH r_contextH, AEGP_RenderLayerContextH l_contextH, AEGP_LayerH *layerPH) ; </pre>
<code>AEGP_GetCompRenderTime</code>	<p>Given the render context, returns the current point in (composition) time to render.</p> <pre> AEGP_GetNthLayerIndexToRender (PR_RenderContextH render_contextH, A_long *time, A_long *time_step) </pre>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
AEGP_GetCompDestination Buffer	<p>Given the render context, returns a buffer in which to place the final rendered output.</p> <pre> AEGP_GetCompToRender(PR_RenderContextH render_contextH, AEGP_CompH compH, PF_EffectWorld *dst); </pre>
AEGP_GetROI	<p>Given the render context provided to the Artisan at render time, returns a handle to the composition.</p> <pre> AEGP_GetROI(PR_RenderContextH render_contextH, A_LegacyRect *roiPR); </pre>
AEGP_RenderLayer	<p>DO NOT USE THIS FUNCTION AFTER VERSION 5.0! Use AEGP_RenderLayerPlus().</p> <p>Renders an AEGP_LayerH into the PF_EffectWorld supplied by the AEGP. This is provided for the rendering of track mattes.</p> <pre> AEGP_RenderLayer(PR_RenderContextH render_contextH, AEGP_LayerH layerH, AEGP_RenderHints render_hints, AEGP_WorldH *render_bufferP); </pre>
AEGP_RenderTexture	<p>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.</p> <pre> 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); </pre> <p>AEGP_RenderHints contains one or more of the following:</p> <pre> AEGP_RenderHints_NONE AEGP_RenderHints_IGNORE_EXTENTS AEGP_RenderHints_NO_TRANSFER_MODE </pre> <p>AEGP_RenderHints_NO_TRANSFER_MODE prevents application of opacity & transfer mode; for use with RenderLayer calls.</p>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
AEGP_DisposeTexture	Disposes of an acquired layer texture. <pre>AEGP_DisposeTexture(PR_RenderContextH render_contextH, AEGP_LayerH layerH, AEGP_WorldH *dst0);</pre>
AEGP_GetFieldRender	Returns the field settings of the given PR_RenderContextH. <pre>AEGP_GetFieldRender(PR_RenderContextH render_contextH, PF_Field *field);</pre>
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. <pre>AEGP_ReportArtisanProgress(PR_RenderContextH render_contextH, A_long countL, A_long totalL);</pre>
AEGP_GetRenderDownsample Factor	Returns the downsample factor of the PR_RenderContextH. <pre>AEGP_GetRenderDownsampleFactor(PR_RenderContextH render_contextH, AEGP_DownsampleFactor *dsfP);</pre>
AEGP_IsBlankCanvas	Determines whether the PR_RenderContextH is blank (empty). <pre>AEGP_IsBlankCanvas(PR_RenderContextH render_contextH, A_Boolean *is_blankPB);</pre>
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. <pre>AEGP_GetRenderLayerToWorldXform(PR_RenderContextH render_contextH, AEGP_RenderLayerContextH layer_contextH, const A_Time *comp_timeP, A_Matrix4 *transform);</pre>
AEGP_GetRenderLayerBounds	Retrieves the bounding rectangle of the layer_contextH (at a given time) within the render_contextH. <pre>AEGP_GetRenderLayerBounds(PR_RenderContextH render_contextH, AEGP_RenderLayerContextH layer_contextH, const A_Time *comp_timeP, A_LegacyRect *boundsP);</pre>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
<code>AEGP_GetRenderOpacity</code>	<p>Returns the opacity of the given layer context at the given time, within the render context.</p> <pre> AEGP_GetRenderOpacity(PR_RenderContextH render_contextH, AEGP_RenderLayerContextH layer_contextH, const A_Time *comp_timePT, A_FpLong *opacityPF); </pre>
<code>AEGP_IsRenderLayerActive</code>	<p>Returns whether or not a given layer context is active within the render context, at the given time.</p> <pre> AEGP_IsRenderLayerActive(PR_RenderContextH render_contextH, AEGP_RenderLayerContextH layer_contextH, const A_Time *comp_timePT, A_Boolean *activePB); </pre>
<code>AEGP_SetArtisanLayerProgress</code>	<p>Sets the progress information for a rendering Artisan. <code>countL</code> is the number of layers completed, <code>num_layersL</code> is the total number of layers the Artisan is rendering.</p> <pre> AEGP_SetArtisanLayerProgress(PR_RenderContextH render_contextH, A_long countL, A_long num_layersL); </pre>
<code>AEGP_RenderLayerPlus</code>	<p>Similar to <code>AEGP_RenderLayer</code>, but takes into account the <code>AEGP_RenderLayerContextH</code>.</p> <pre> AEGP_RenderLayerPlus(PR_RenderContextH r_contextH, AEGP_LayerH layerH, AEGP_RenderLayerContextH l_contextH, AEGP_RenderHints render_hints, AEGP_WorldH *bufferP); </pre>
<code>AEGP_GetTrackMatteContext</code>	<p>Retrieves the <code>AEGP_RenderLayerContextH</code> for the specified render and fill contexts.</p> <pre> AEGP_GetTrackMatteContext(PR_RenderContextH render_contextH, AEGP_RenderLayerContextH fill_contextH, AEGP_RenderLayerContextH *mattePH); </pre>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
AEGP_RenderTextureWithReceipt	<p>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.</p> <pre> 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); </pre>
AEGP_GetNumberOfSoftwareEffects	<p>Returns the number of software effects applied in the given AEGP_RenderLayerContextH.</p> <pre> AEGP_GetNumberOfSoftwareEffects(PR_RenderContextH ren_contextH, AEGP_RenderLayerContextH lyr_contextH, A_short *num_sft_FXPS); </pre>
AEGP_RenderLayerPlusWithReceipt	<p>An improvement over AEGP_RenderLayerPlus, this function also provides an AEGP_RenderReceiptH for caching purposes.</p> <pre> 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); </pre>
AEGP_DisposeRenderReceipt	<p>Frees an AEGP_RenderReceiptH.</p> <pre> AEGP_DisposeRenderReceipt(AEGP_RenderReceiptH receiptH); </pre>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
<code>AEGP_CheckRenderReceipt</code>	<p>Checks with After Effects' internal caching to determine whether a given <code>AEGP_RenderReceiptH</code> is still valid. The <code>num_effectsS</code> parameter is new in 7.0.</p> <pre> 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); </pre>
<code>AEGP_GenerateRenderReceipt</code>	<p>New in 7.0 Generates a <code>AEGP_RenderReceiptH</code> for a layer as if the first <code>num_effectsS</code> have been rendered</p> <pre> AEGP_GenerateRenderReceipt(PR_RenderContextH current_contextH, AEGP_RenderLayerContextH current_lyr_contextH, AEGP_NumEffectsToRenderType num_effectsS, AEGP_RenderReceiptH *render_receiptPH); </pre>
<code>AEGP_GetNumBinsToRender</code>	<p>Returns the number of bins After Effects wants the artisan to render.</p> <pre> AEGP_GetNumBinsToRender(const PR_RenderContextH contextH, A_long *num_binsPL); </pre>
<code>AEGP_SetNthBin</code>	<p>Sets the given render context to be the n-th bin to be rendered by After Effects.</p> <pre> AEGP_SetNthBin(const PR_RenderContextH contextH, A_long n); </pre>
<code>AEGP_GetBinType</code>	<p>Retrieves the type of the given bin.</p> <pre> AEGP_GetBinType(const PR_RenderContextH contextH, AEGP_BinType *bin_typeP); </pre> <p><code>AEGP_BinType</code> will be one of the following:</p> <pre> AEGP_BinType_NONE AEGP_BinType_2D AEGP_BinType_3D </pre>

TABLE 99: AEGP_CANVASSUITE7

Function	Purpose
AEGP_GetRenderLayerToWorldXform2D3D	<p>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.</p> <pre> AEGP_GetRenderLayerToWorldXform2D3D(PR_RenderContextH render_contextH, AEGP_RenderLayerContextHlayer_contextH, const A_Time *comp_timeP, A_Boolean only_2dB, A_Matrix4 *transformP); </pre>
<i>Functions below are for interactive artisans only</i>	
AEGP_GetPlatformWindowRef	<p>Retrieves the platform-specific window context into which to draw the given PR_RenderContextH.</p> <pre> AEGP_GetPlatformWindowRef(const PR_RenderContextH contextH, AEGP_PlatformWindowRef *window_refP); </pre>
AEGP_GetViewportScale	<p>Retrieves the source-to-frame downsample factor for the given PR_RenderContextH.</p> <pre> AEGP_GetViewportScale(const PR_RenderContextH contextH, A_FpLong *scale_xPF, A_FpLong *scale_yPF); </pre>
AEGP_GetViewportOrigin	<p>Retrieves to origin of the source, within the frame (necessary to translate between the two), for the given PR_RenderContextH.</p> <pre> AEGP_GetViewportOrigin(const PR_RenderContextH contextH, A_long *origin_xPL, A_long *origin_yPL); </pre>
AEGP_GetViewportRect	<p>Retrieves the bounding rectangle for the area to be drawn, for the given PR_RenderContextH.</p> <pre> AEGP_GetViewportRect(const PR_RenderContextH contextH, A_LegacyRect *v_rectPR); </pre>
AEGP_GetFallowColor	<p>Retrieves the color used for the fallow regions in the given PR_RenderContextH.</p> <pre> AEGP_GetFallowColor(const PR_RenderContextH contextH, PF_Pixel8 *fallow_colorP); </pre>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
AEGP_GetInteractiveCheckerboard	Retrieves whether or not the checkerboard is currently active for the given PR_RenderContextH. <pre>AEGP_GetInteractiveCheckerboard(const PR_RenderContextH contextH, A_Boolean *cboard_onPB);</pre>
AEGP_GetInteractiveCheckerboardColors	Retrieves the colors used in the checkerboard. <pre>AEGP_GetInteractiveCheckerboardColors(const PR_RenderContextH contextH, PF_Pixel *color1P, PF_Pixel *color2P);</pre>
AEGP_GetInteractiveCheckerboardSize	Retrieves the width and height of one checkerboard square. <pre>AEGP_GetInteractiveCheckerboardSize(const PR_RenderContextH contextH, A_u_long *cbd_widthPLu, A_u_long *cbd_heightPLu);</pre>
AEGP_GetInteractiveCachedBuffer	Retrieves the cached AEGP_WorldH last used for the PR_RenderContextH. <pre>AEGP_GetInteractiveCachedBuffer(const PR_RenderContextH contextH, AEGP_WorldH *buffer);</pre>
AEGP_ArtisanMustRenderAsLayer	Determines whether or not the artisan must render the current AEGP_RenderLayerContextH as a layer. <pre>AEGP_ArtisanMustRenderAsLayer(const PR_RenderContextH contextH, AEGP_RenderLayerContextH layer_contextH, A_Boolean *use_texturePB);</pre>
AEGP_GetInteractiveDisplayChannel	Returns which channels should be displayed by the interactive artisan. <pre>AEGP_GetInteractiveDisplayChannel(const PR_RenderContextH contextH, AEGP_DisplayChannelType *channelP);</pre> <p>AEGP_DisplayChannelType will be one of the following:</p> <pre>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</pre>

TABLE 99: AEGP_CANVASUITE7

Function	Purpose
AEGP_GetInteractiveExposure	Returns the exposure for the given PR_RenderContextH, expressed as a floating point number. <pre>AEGP_GetInteractiveExposure(const PR_RenderContextH rcH, A_FpLong *exposurePF);</pre>
AEGP_GetColorTransform	New in CS3. Returns the color transform for the given PR_RenderContextH. <pre>AEGP_GetColorTransform)(const PR_RenderContextH render_contextH, A_Boolean *cms_onB, A_u_long *xform_keyLu, void *xformP);</pre>
AEGP_GetCompShutterTime	New in CS5. Returns the shutter angle for the given PR_RenderContextH. <pre>AEGP_GetCompShutterTime)(PR_RenderContextH render_contextH, A_Time *shutter_time, A_Time *shutter_dur);</pre>

CONVERT BETWEEN DIFFERENT CONTEXTS

Convert from one context to another.

TABLE 100: AEGP_ARTISANUTILSUITE1

Function	Purpose
AEGP_GetGlobalContextFromInstanceContext	Given an instance context, returns a handle to the global context. <pre>AEGP_GetGlobalContextFromInstanceContext(const PR_InstanceContextH instance_contextH, PR_GlobalContextH *global_contextPH);</pre>
AEGP_GetInstanceContextFromRenderContext	Given the render context, returns a handle to the instance context. <pre>AEGP_GetInstanceContextFromRenderContext(const PR_RenderContextH render_contextH, PR_InstanceContextH *instnc_ctxtPH);</pre>
AEGP_GetInstanceContextFromQueryContext	Given a query context, returns a handle to the instance context. <pre>AEGP_GetInstanceContextFromQueryContext(const PR_QueryContextH query_contextH, PR_InstanceContextH *instnce_contextPH);</pre>

TABLE 100: AEGP_ARTISANUTILSUITE1

Function	Purpose
<code>AEGP_GetGlobalData</code>	Given the global context, returns a handle to global data. <pre>AEGP_GetGlobalData(const PR_GlobalContextH global_contextH, PR_GlobalDataH *global_dataPH);</pre>
<code>AEGP_GetInstanceData</code>	Given an instance context, return the associated instance data. <pre>AEGP_GetInstanceData(const PR_InstanceContextH instance_contextH, PR_InstanceDataH *instance_dataPH);</pre>
<code>AEGP_GetRenderData</code>	Given a render context, returns the associated render data. <pre>AEGP_GetRenderData(const PR_RenderContextH render_contextH, PR_RenderDataH *render_dataPH);</pre>

TRACK MATTES AND TRANSFORM FUNCTIONS

Use the `AEGP_CompositeSuite` to copy pixel worlds, operate on track mattes, and apply transfer functions.

TABLE 101: AEGP_COMPOSITESUITE2

Function	Purpose
<code>AEGP_ClearAlphaExceptRect</code>	For the given <code>PF_EffectWorld</code> , sets the alpha to fully transparent except for the specified rectangle. <pre>AEGP_ClearAlphaExceptRect(A_Rect *clipped_dst_rectPR, PF_EffectWorld *dstP);</pre>
<code>AEGP_PrepTrackMatte</code>	Mattes the pixels in a <code>PF_EffectWorld</code> with the <code>PF_Pixel</code> described in <code>src_masks</code> , putting the output into an array of pixels <code>dst_mask</code> . NOTE: Unlike most of the other pixel mangling functions provided by After Effects, this one doesn't take <code>PF_EffectWorld</code> arguments; rather, you can simply pass the data pointer from within the <code>PF_EffectWorld</code> . This can be confusing, but as a bonus, the function pads output appropriately so that <code>num_pix</code> pixels are always output. <pre>AEGP_PrepTrackMatte(A_long num_pix, A_Boolean deepB, const PF_Pixel *src_mask, PF_MaskFlags mask_flags, PF_Pixel *dst_mask);</pre>

TABLE 101: AEGP_COMPOSITE SUITE 2

Function	Purpose
AEGP_GetTransferDesc	<p>No longer supported in 7.0 and later.</p> <p>Given a compositing mode, quality, and mode flags (which indicate whether or not the alpha is inverted), populates an AEGP_TransferDesc (described below).</p> <pre> AEGP_GetTransferDesc(const PF_CompositeMode *c_mode, PF_Quality quality, PF_ModeFlags m_flags, AEGP_TransferDesc *descP); </pre> <pre> typedef struct { AEGP_TransPixFuncpix_funcB; A_Boolean has_neutralB; A_Boolean needs_extensionB; A_Boolean std_rgb_onlyB; A_Boolean combine_alphaB; AEGP_MatteFunc matte_func; AEGP_MatteFunc dematte_func; } AEGP_TransferDesc; </pre>
AEGP_DescriptionComposite	<p>No longer supported in 7.0 and later.</p> <p>Uses a AEGP_TransferDesc to composite num_pix pixels from src_pix to dst_pix, using mask_pix.</p> <pre> AEGP_DescriptionComposite(const AEGP_TransferDesc *dsc, A_long num_pix, A_u_char *scratch_alpha, const PF_CompositeMode *cmode, const PF_Pixel *src_pix, const PF_Pixel *mask_pix, PF_Pixel *dst_pix); </pre>

TABLE 101: AEGP_COMPOSITESUITE2

Function	Purpose
AEGP_TransferRect	<p>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.</p> <pre> 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); </pre>
AEGP_CopyBits_LQ	<p>Copies a rectangle of pixels (pass a NULL rectangle to get all pixels) from one PF_EffectWorld to another, at low quality.</p> <pre> AEGP_CopyBits_LQ(PF_EffectWorld *src_worldP, A_Rect *src_r, A_Rect *dst_r, PF_EffectWorld *dst_worldP); </pre>
AEGP_CopyBits_HQ_Straight	<p>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.</p> <pre> AEGP_CopyBits_HQ_Straight(PF_EffectWorld *src, A_Rect *src_r, A_Rect *dst_r, PF_EffectWorld *dst); </pre>
AEGP_CopyBits_HQ_Premul	<p>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.</p> <pre> AEGP_CopyBits_HQ_Premul(PF_EffectWorld *src, A_Rect *src_r, A_Rect *dst_r, PF_EffectWorld *dst); </pre>

SMILE! CAMERAS

Obtains the camera geometry, including camera properties (type, lens, depth of field, focal distance, aperture, et cetera).

TABLE 102: AEGP_CAMERA_SUITE2

Function	Purpose
AEGP_GetCamera	<p>Given a layer handle and time, returns the current camera layer handle.</p> <pre> AEGP_GetCamera(PR_RenderContextH render_contextH, const A_Time *comp_timeP, AEGP_LayerH *camera_layerPH); </pre>
AEGP_GetCameraType	<p>Given a layer, returns the camera type of the layer.</p> <pre> AEGP_GetCameraType(AEGP_LayerH aegp_layerH, AEGP_CameraType *camera_typeP; </pre> <p>The camera type can be the following:</p> <pre> AEGP_CameraType_NONE = -1 AEGP_CameraType_PERSPECTIVE AEGP_CameraType_ORTHOGRAPHIC </pre>
AEGP_GetDefaultCameraDistanceToImagePlane	<p>Given a composition handle, returns the camera distance to the image plane.</p> <pre> AEGP_GetDefaultCameraDistanceToImagePlane(AEGP_CompH compH, A_FpLong *dist_to_planePF) </pre>
AEGP_GetCameraFilmSize	<p>Retrieves the size (and units used to measure that size) of the film used by the designated camera.</p> <pre> AEGP_GetCameraFilmSize(AEGP_LayerH camera_layerH, AEGP_FilmSizeUnits *film_size_unitsP, A_FpLong *film_sizePF0); </pre>
AEGP_SetCameraFilmSize	<p>Sets the size (and unites used to measure that size) of the film used by the designated camera.</p> <pre> AEGP_SetCameraFilmSize)(AEGP_LayerH camera_layerH, AEGP_FilmSizeUnits film_size_units, A_FpLong *film_sizePF0); </pre>

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;

$\Theta = 1/2$ field of view

$\tan(\Theta) = 1/2$ composition height / focal length

focal length = $2 \tan(\Theta)$ / composition height

HIT THE LIGHTS!

Get and set the type of lights in a composition.

TABLE 103: AEGP_LIGHTSUITE2

Function	Purpose
<code>AEGP_GetLightType</code>	<p>Retrieves the <code>AEGP_LightType</code> of the specified camera layer.</p> <pre>AEGP_GetLightType(AEGP_LayerH light_layerH, AEGP_LightType *light_typeP);</pre> <p><code>AEGP_LightType</code> will be one of the following:</p> <p><code>AEGP_LightType_PARALLEL</code> <code>AEGP_LightType_SPOT</code> <code>AEGP_LightType_POINT</code> <code>AEGP_LightType_AMBIENT</code></p>
<code>AEGP_SetLightType</code>	<p>Sets the <code>AEGP_LightType</code> for the specified camera layer.</p> <pre>AEGP_SetLightType(AEGP_LayerH light_layerH, AEGP_LightType light_type);</pre>

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 $S * \text{power}(H \cdot N, \text{shine})$, 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.

TRANSFORM CONVENTIONS

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 they'll need in order to respond appropriately to the various queries After Effects will send to their [PR_QueryFunc](#) 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 104: AEGP_QUERYXFORMSUITE2

Function	Purpose
<code>AEGP_QueryXformGetSrcType</code>	<p>Given a query context, returns transform source currently being modified.</p> <pre>AEGP_QueryXformGetSrcType(PR_QueryContextH query_contextH, AEGP_QueryXformType *src_type);</pre> <p>The query context will be one of the following:</p> <pre>AEGP_Query_Xform_LAYER, AEGP_Query_Xform_WORLD, AEGP_Query_Xform_VIEW, AEGP_Query_Xform_SCREEN</pre>
<code>AEGP_QueryXformGetDstType</code>	<p>Given a query context, returns the currently requested transform destination.</p> <pre>AEGP_QueryXformGetDstType(PR_QueryContextH query_contextH, AEGP_QueryXformType *dst_type);</pre>
<code>AEGP_QueryXformGetLayer</code>	<p>Used if the source or destination type is a layer. Given a query context, returns the layer handle.</p> <pre>AEGP_QueryXformGetLayer(PR_QueryContextH query_contextH, AEGP_LayerH *layerPH);</pre>

TABLE 104: AEGP_QUERYXFORMSUITE2

Function	Purpose
AEGP_QueryXformGetComp	Given a query context, returns the current composition handle. <pre>AEGP_QueryXformGetComp(PR_QueryContextH query_contextH, AEGP_CompH *compPH);</pre>
AEGP_QueryXformGetTransformTime	Given a query context, returns the time of the transformation. <pre>AEGP_QueryXformGetTransformTime(PR_QueryContextH query_contextH, A_Time *time);</pre>
AEGP_QueryXformGetViewTime	Given a query context, returns the time of the associated view. <pre>AEGP_QueryXformGetViewTime(PR_QueryContextH query_contextH, A_Time *time);</pre>
AEGP_QueryXformGetCamera	Given a query context, returns the current camera layer handle. <pre>AEGP_QueryXformGetCamera(PR_QueryContextH query_contextH, AEGP_LayerH *camera_layerPH);</pre>
AEGP_QueryXformGetXform	Given a query context, returns the current matrix transform. <pre>AEGP_QueryXformGetXform(PR_QueryContextH query_contextH, A_Matrix4 *xform);</pre>
AEGP_QueryXformSetXform	Given a query context, return the matrix transform you compute in xform. <pre>AEGP_QueryXformSetXform(PR_QueryContextH query_contextH, A_Matrix4 *xform);</pre>
AEGP_QueryWindowRef	Sets the window reference to be used (by After Effects) for the given PR_QueryContextH. <pre>AEGP_QueryWindowRef(PR_QueryContextH q_contextH, AEGP_PlatformWindowRef *window_refP);</pre>
AEGP_QueryWindowClear	Returns which AEGP_PlatformWindowRef (and A_Rect) to clear, for the given PR_QueryContextH. <pre>AEGP_QueryWindowClear(PR_QueryContextH q_contextH, AEGP_PlatformWindowRef *window_refP, A_LegacyRect *boundsPR);</pre>

TABLE 104: AEGP_QUERYXFORMSUITE2

Function	Purpose
AEGP_QueryFrozenProxy	<p>Returns whether or not the textures used in the given PR_QueryContextH should be frozen.</p> <pre> AEGP_QueryFrozenProxy(PR_QueryContextH q_contextH, A_Boolean *onPB); </pre>
AEGP_QuerySwapBuffer	<p>Sent after rendering and camera/light handle drawing is complete; After Effects returns the buffer into which the artisan should draw its output.</p> <pre> AEGP_QuerySwapBuffer(PR_QueryContextH q_contextH, AEGP_PlatformWindowRef *window_refP, AEGP_WorldH *dest_bufferP); </pre>
AEGP_QueryDrawProcs	<p>Sets the interactive drawing functions After Effects will call while drawing camera and lighting handles into the artisan's provided context.</p> <pre> AEGP_QueryDrawProcs(PR_QueryContextH query_contextH, PR_InteractiveDrawProcs *window_refP); </pre>
AEGP_QueryPrepareForLineDrawing	<p>Informs After Effects about the context into which it will be drawing.</p> <pre> AEGP_QueryPrepareForLineDrawing(PR_QueryContextH query_contextH, AEGP_PlatformWindowRef *window_refP, A_LegacyRect *viewportP, A_LPoint *originP, A_FloatPoint *scaleP); </pre>
AEGP_QueryUnprepareForLineDrawing	<p>As far as After Effects is concerned, the artisan is done drawing lines.</p> <pre> AEGP_QueryUnprepareForLineDrawing(PR_QueryContextH query_contextH, AEGP_PlatformWindowRef *window_refP); </pre>

INTERACTIVE DRAWING FUNCTIONS

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 105: 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\(\)](#). `AEGP_QueryXformGetViewTime()` gets the time of the view, which is used in calling [AEGP_GetLayerToWorldXformFromView\(\)](#).

9 • AEIOs

AEGBs which perform file input and output ('AEIOs') manage all file handling for file types not handled by After Effects by supplying functions matching certain prototypes, which After Effects hooks into its internal messaging. This new API replaces and supersedes Photoshop Format ('FXIF') plug-ins.

WHAT'S NEW IN CS5?

We have added [AEGP_IOOutSuite4](#) to use Unicode platform paths. The new [AEIO_FunctionBlock4](#) replaces earlier versions. `AEIO_InitInSpecFromFile`, `AEIO_SetOutputFile`, and `AEIO_VerifyFileImportable`, which previously took `char *` as input, now take `const A_UTF16Char *`. `AEIO_GetNthAuxFileSpec`, now takes and fills in a `AEGB_MemHandle`, which the caller must dispose. `AEIO_FunctionBlock1,2,3` are gone, but you can still leave functions at the end NULL (all the ones past `AEIO_UserAudioOptionsDialog`).

WHAT'S NEW IN CS3 (8.0)?

`AEGB_SetInSpecCMOptions` was added to [AEGP_IOInSuite2](#) to allow AEIOs to indicate desired color management behavior. Also, calls were added to [AEGP_IOOutSuite3](#) to let AEIOs access color profiles, as required.

After Effects now 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 SDK](#). 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. However, MediaCore importers cannot defer file import to an AEIO.

WHAT CAN AEIOS DO?

AEIOs do everything for file of a given type that After Effects (or the plug-ins which ship with After Effects) would normally do; open existing files, manage file-specific interpretation options, provide audio and frames from the file to After Effects in

AEGP_SoundWorld and PF_EffectWorld format, create and manage output options for render queue items, create output files and saving frames (provided by After Effects as PF_EffectWorlds) into those files. Additionally, AEIOs can create files interactively, asking users for the settings they'd like instead of reading them from a file.

WHAT THEY CANNOT DO

AEIOs cannot take over file handling for any file type for which After Effects already provides a plug-in. Instead, you need to develop a MediaCore importer plug-in, using the [Premiere Pro SDK](#). AEIOs also do not participate in the MediaCore importer priority system. So if you use a MediaCore importer plug-in to override file handling by a built-in AEIO, you will not be able to defer unsupported files back to the AEIO to handle files not supported by your importer.

AEIO, OR AEGP?

AEIOs provide pixels and audio data to After Effects. If you're writing an importer 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 submenu.

HOW THEY DO IT

From within their 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\(\)](#).

AEIO_MODULEINFO

Let's investigate this structure further. 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.

As of CS4, file extensions cannot be more than three characters long, even though we have a few built-in importers with longer extensions.

TABLE 106: 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 what sig you're using.
name	Descriptive name for your AEIO plug-in.
flags	Set of AEIO_ModuleFlags .
flags2	Set of AEIO_ModuleFlags2 .
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.

TABLE 106: AEIO_MODULEINFO

Member	Purpose
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.

BEHAVIOR FLAGS

AEIOs set flags (like effect plug-ins use global outflags) to indicate their behavior to After Effects.

TABLE 107: AEIO_MODULEFLAGS

Flag	Purpose
AEIO_MFlag_INPUT	AEIO is an input module.
AEIO_MFlag_OUTPUT	AEIO is an output module (one plug-in can be both).
AEIO_MFlag_FILE	Each clip imported directly corresponds to a file, somewhere.
AEIO_MFlag_STILL	Supports still images, not video.
AEIO_MFlag_VIDEO	Supports video images, not stills.
AEIO_MFlag_AUDIO	Supports audio.
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.
AEIO_MFlag_INTERACTIVE_GET	A new input sequence necessitates user interaction. This would be the case for a non-file-based input module.
AEIO_MFlag_INTERACTIVE_PUT	A new output sequence necessitates user interaction. This would be the case for a non-file-based output module.
AEIO_MFlag_CANT_CLIP	The AEIO's drawing functions cannot accept dimensions smaller than the requested dimensions.

TABLE 107: AEIO_MODULEFLAGS

Flag	Purpose
AEIO_MFlag_MUST_INTERACT_PUT	The AEIO must display a dialog box, even if a valid options data handle is available.
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).
AEIO_MFlag_CAN_ADD_FRAMES_NON_LINEAR	The AEIO supports adding non-sequential frames.
AEIO_MFlag_HOST_DEPTH_DIALOG	The AEIO wants After Effects to display a bit-depth selection dialog.
AEIO_MFlag_HOST_FRAME_START_DIALOG	The AEIO wants After Effects to display a dialog requesting that the user specify a starting frame.
AEIO_MFlag_RESERVED1	
AEIO_MFlag_NO_OPTIONS	The AEIO does not accept output options.
AEIO_MFlag_RESERVED2	
AEIO_MFlag_RESERVED3	
AEIO_MFlag_NO_PIXELS	The AEIO's file format doesn't actually store pixels.
AEIO_MFlag_SEQUENCE_OPTIONS_OK	The AEIO will adopt the sequence options of its parent if a folder is selected.
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).
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.
AEIO_MFlag_HAS_LAYERS	The AEIO supports multiple layers in a single document.
AEIO_MFlag_SCRAP	The AEIO has a clipboard parsing component.
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)
AEIO_MFlag_SEQ_OPTIONS_DLG	The AEIO has sequence options.
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.

TABLE 107: AEIO_MODULEFLAGS

Flag	Purpose
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.
AEIO_MFlag_CAN_DO_MARKERS	The file format support by the AEIO supports markers, url flips, and/or chapters.
AEIO_MFlag_CAN_DRAW_DEEP	The AEIO can draw into 16bpc (“deep”) PF_EffectWorlds.
AEIO_MFlag_RESERVED4	Special super-secret flag. Doesn’t do anything...or does it? (No, it <i>doesn’t</i> .)

AEIO_MODULEFLAGS2

Gotta have dem flags...

TABLE 108: AEIO_MODULEFLAGS2

Flag	Purpose
AEIO_MFlag2_AUDIO_OPTIONS	The AEIO has an audio options dialog.
AEIO_MFlag2_SUPPORTS_QUERY_DYNAMIC_FLAGS	The AEIO supports the dynamic querying of module flags.
AEIO_MFlag2_SEND_ADDMARKER_BEFORE_ADDFRAME	The AEIO wants to receive marker data before outputting video or audio (useful for MPEG streams).
AEIO_MFlag2_CAN_DO_MARKERS_2	The AEIO supports combined markers; URL flips, chapters, and comments.
AEIO_MFlag2_CAN_DRAW_FLOAT	New in 7.0. The AEIO can draw into float (32-bpc) worlds.
AEIO_MFlag2_CAN_DO_AUDIO_32	New in 7.0. Supports 32-bit audio output.

NEW KIDS ON THE FUNCTION BLOCK

During its main entry point function, each AEIO plug-in must fill in an AEIO_FunctionBlock, indicating the names of the functions After Effects shall call for

different file-related tasks. You can often invoke “best-case” behavior by having After Effects handle the call for you (by returning `AEIO_Err_USE_DFLT_CALLBACK`).

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
<code>AEIO_InitInSpecFromFile</code>	<p>Given a file path, describe its contents to After Effects in the provided <code>AEIO_InSpecH</code>. Use all appropriate “set” calls from the AEIO_AEGPIOInSuite 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.</p> <p>The file path is a NULL-terminated UTF-16 string with platform separators.</p> <pre>AEIO_InitInSpecFromFile(AEIO_BasicData *basic_dataP, const A_UTF16Char *file_pathZ, AEIO_InSpecH inH);</pre>
<code>AEIO_InitInSpecInteractive</code>	<p>Using some form of user interaction (and not a file path provided by After Effects), describe the audio and video your generated <code>AEIO_InSpecH</code> contains.</p> <pre>AEIO_InitInSpecInteractive(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH);</pre>
<code>AEIO_DisposeInSpec</code>	<p>Free an <code>AEIO_InSpecH</code>.</p> <pre>AEIO_DisposeInSpec(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH);</pre>
<code>AEIO_FlattenOptions</code>	<p>For the given <code>AEIO_InSpecH</code>, return a flattened version of the data contained in its options handle. Obtain the unflattened options handle using AEGP_GetInSpecOptionsHandle.</p> <pre>AEIO_FlattenOptions(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, AEIO_Handle *flat_optionsPH);</pre>
<code>AEIO_InflateOptions</code>	<p>For the given <code>AEIO_InSpecH</code>, create (using AEGP_SetInSpecOptionsHandle) an unflattened version of its flattened option data.</p> <pre>AEIO_InflateOptions(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, AEIO_Handle flat_optionsH);</pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_SynchInSpec	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Inspect the AEIO_InSpecH, update its options if necessary), and indicate whether or not you made changes.</p> <pre>AEIO_SynchInSpec(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, A_Boolean *changed0);</pre>
AEIO_GetActiveExtent	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Populate the provided A_LRect with the active extent of the file's pixels at the given time.</p> <pre>AEIO_GetActiveExtent(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, const A_Time *tr, A_LRect *extent);</pre>
AEIO_GetInSpecInfo	<p>Provide a textual description of the AEIO_InSpecH in the provided AEIO_Verbiage.</p> <pre>AEIO_GetInSpecInfo(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, AEIO_Verbiage *verbiageP);</pre> <p>This function gets called OFTEN; every time we refresh the project window. Keep allocations to a minimum. In the AEIOs that ship with After Effects, we check for a valid optionsH (using AEGP_GetInSpecOptionsHandle); 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 <i>will</i> 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).</p>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_DrawSparseFrame	<p>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.</p> <pre> 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); </pre> <p>NOTE: return data as linear light (1.0), and After Effects will perform any necessary transformations to bring the footage into the working colorspace.</p>
AEIO_GetDimensions	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Provide the dimensions (and, if necessary, scaling factor) of the video in the AEIO_InSpecH.</p> <pre> AEIO_GetDimensions(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, const AEIO_RationalScale *rs0, A_long *width0, A_long *height0); </pre>
AEIO_GetDuration	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Provide the duration of an AEIO_InSpecH.</p> <pre> AEIO_GetDuration(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, A_Time *trP); </pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_GetTime	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Provide the timebase of an AEIO_InSpecH.</p> <pre> AEIO_GetTime(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, A_Time *tr); </pre> <p>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;</p>
AEIO_GetSound	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Provide sound from an AEIO_InSpecH, in the format indicated in the AEIO_GetSoundPB*.</p> <pre> AEIO_GetSound(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, const AEIO_GetSoundPB *gs_pbP, AEIO_SndWorldH worldH); </pre>
AEIO_InqNextFrameTime	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Provide the time of the next frame (in the source footage's timebase) within the AEIO_InSpecH.</p> <pre> 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); </pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_InitOutputSpec	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Perform any initialization necessary for a new AEIO_OutSpecH, and indicate whether you made changes.</p> <pre>InitOutputSpec(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_Boolean *user_interacted);</pre> <p>NOTE: The first time your AEIO is used, After Effects caches the last-known-good optionsH in its preferences. When testing this function, delete your preferences (by holding down Alt or option during launch) often.</p>
AEIO_GetFlatOutputOptions	<p>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.</p> <pre>AEIO_GetFlatOutputOptions(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, AEIO_Handle *optionsH);</pre>
AEIO_DisposeOutputOptions	<p>Free an AEIO_OutSpecH.</p> <pre>AEIO_DisposeOutputOptions(AEIO_BasicData *basic_dataP, void *optionsPV);</pre>
AEIO_UserOptionsDialog	<p>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.</p> <pre>AEIO_UserOptionsDialog(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, PF_EffectWorld *sample0, A_Boolean *interacted0);</pre>
AEIO_GetOutputInfo	<p>Describe (in text) the output options in an AEIO_OutSpecH.</p> <pre>AEIO_GetOutputInfo(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, AEIO_Verbiage *verbiage);</pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_OutputInfoChanged	<p>Update the AEIO_OutSpecH based on the current settings (using the various Get functions to obtain them).</p> <pre>AEIO_OutputInfoChanged(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH);</pre>
AEIO_SetOutputFile	<p>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.</p> <p>The file path is a NULL-terminated UTF-16 string with platform separators.</p> <pre>AEIO_SetOutputFile(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_UTF16Char *file_pathZ);</pre>
AEIO_StartAdding	<p>Prepare to add frames to the output file. Note: this is your first opportunity to allocate pixel buffers based on valid output spec values.</p> <pre>AEIO_StartAdding(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_long flags);</pre>
AEIO_AddFrame	<p>Add frame(s) to output file. You may pass a pointer to a function you want called if the user interrupts the render.</p> <pre>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);</pre>
AEIO_EndAdding	<p>Perform any clean-up associated with adding frames.</p> <pre>AEIO_EndAdding(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_long flags);</pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_OutputFrame	<p>Output a single frame.</p> <pre>AEIO_OutputFrame(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, PF_EffectWorld *wP);</pre>
AEIO_WriteLabels	<p>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.</p> <pre>AEIO_WriteLabels(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, AEIO_LabelFlags *written);</pre>
AEIO_GetSizes	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Provide information about file size and remaining free space on output volume.</p> <pre>AEIO_GetSizes(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_u_longlong *free_space, A_u_longlong *file_size);</pre>
AEIO_Flush	<p>Destroy any options or user data associated with the OutSpecH.</p> <pre>AEIO_Flush(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH);</pre>
AEIO_AddSoundChunk	<p>Add the given sound to the output file.</p> <pre>AEIO_AddSoundChunk(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, const A_Time *start, AEIO_SndWorldH swH);</pre>
AEIO_Idle	<p>Optional. Do something with idle time. AEIO_Err_USE_DFLT_CALLBACK is not supported.</p> <pre>AEIO_Idle(AEIO_BasicData *basic_dataP, AEIO_ModuleSignature sig, AEIO_IdleFlags *idle_flags0);</pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_GetDepths	<p>Set AEIO_OptionsFlags to indicate which pixel and color depths are valid for your output format.</p> <pre>AEIO_GetDepths(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, AEIO_OptionsFlags *which);</pre>
AEIO_GetOutputSuffix	<p>AEIO_Err_USE_DFLT_CALLBACK allowed. Describe the three character extension for the output file.</p> <pre>AEIO_GetOutputSuffix(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_char *suffix);</pre>
AEIO_SeqOptionsDlg	<p>Display a footage options dialog, and indicate whether the user made any changes.</p> <pre>AEIO_SeqOptionsDlg(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, A_Boolean *interactedPB);</pre>
AEIO_GetNumAuxChannels	<p>Enumerate the auxiliary (beyond red, green, blue and alpha) channels of data contained in an AEIO_InSpecH.</p> <pre>AEIO_GetNumAuxChannels(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, A_long *num_channelsPL);</pre>
AEIO_GetAuxChannelDesc	<p>Describe the data type, name, channel, and dimensionality of an auxiliary data channel.</p> <pre>AEIO_GetAuxChannelDesc(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, long chan_indexL, PF_ChannelDesc *descP);</pre>
AEIO_DrawAuxChannel	<p>Draw the auxiliary channel(s) from an AEIO_InSpecH.</p> <pre>AEIO_DrawAuxChannel(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, A_long chan_indexL, const AEIO_DrawFramePB *pbP, PF_ChannelChunk *chunkP);</pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_FreeAuxChannel	Free data associated with an auxiliary channel. <pre>AEIO_FreeAuxChannel(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, PF_ChannelChunk *chunkP);</pre>
AEIO_NumAuxFiles	Enumerate the files needed to render the given AEIO_InSpecH. <pre>AEIO_NumAuxFiles(AEIO_BasicData *basic_dataP, AEIO_InSpecH seqH, A_long *files_per_framePL);</pre>
AEIO_GetNthAuxFileSpec	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. <pre>AEIO_GetNthAuxFileSpec(AEIO_BasicData *basic_dataP, AEIO_InSpecH seqH, A_long frame_numL, A_long n, AEGP_MemHandle *pathPH);</pre>
AEIO_CloseSourceFiles	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. <pre>AEIO_CloseSourceFiles(AEIO_BasicData *basic_dataP, AEIO_InSpecH seqH, A_Boolean closeB);</pre> <p>TRUE for close, FALSE for open.</p>
AEIO_CountUserData	Enumerate the units of user data associated with the AEIO_InSpecH. <pre>AEIO_CountUserData(AEIO_BasicData *basic_dataP, AEIO_InSpecH inH, A_u_long typeLu, A_u_long max_sizeLu, A_u_long *num_of_typePLu);</pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_SetUserData	<p>Set user data (of the given index and type) for the given AEIO_OutSpecH.</p> <pre>AEIO_SetUserData(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outh, A_u_long typeLu, A_u_long indexLu, const AEIO_Handle dataH);</pre>
AEIO_GetUserData	<p>Describe the user data (at the index and of the type given) associated with the AEIO_InSpecH.</p> <pre>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);</pre>
AEIO_AddMarker	<p>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.</p> <pre>AEIO_AddMarker(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outh, A_long frame_index, AEIO_MarkerType marker_type, void *marker_dataPV, AEIO_InterruptFuncs *inter0);</pre>
AEIO_VerifyFileImportable	<p>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.</p> <p>The file path is a NULL-terminated UTF-16 string with platform separators.</p> <pre>AEIO_VerifyFileImportable(AEIO_BasicData *basic_dataP, AEIO_ModuleSignature sig, const A_UTF16Char *file_pathZ, A_Boolean *importablePB);</pre>

TABLE 109: AEIO_FUNCTIONBLOCK4

Function	Response
AEIO_UserAudioOptionsDialog	Display an audio options dialog. <pre>AEIO_UserAudioOptionsDialog(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_Boolean *interacted0);</pre>
AEIO_AddMarker2	New in 7.0. Add a marker. This has been replaced by AEIO_AddMarker3. <pre>AEIO_AddMarker2(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_long frame_index, const AEIO_Marker *markerP, AEIO_InterruptFuncs *inter0);</pre>
AEIO_AddMarker3	New in CS3 (8.0). Add a marker, with a flag specifying whether or not this is a composition marker. <pre>AEIO_AddMarker3(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_long frame_index, AEGP_ConstMarkerValP marker_valP, AEIO_RenderMarkerFlag marker_flag, AEIO_InterruptFuncs *inter0);</pre>
AEIO_GetMimeType	New in CS4. Describe the output mime type. This is used for XMP support. <pre>AEIO_GetMimeType(AEIO_BasicData *basic_dataP, AEIO_OutSpecH outH, A_long mime_type_sizeL, char *mime_typeZ);</pre>

WHAT GOES IN

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 110: AEGPIOINSUITE4

Function	Purpose
AEGP_GetInSpecOptionsHandle	Retrieves the options data (created by your AEIO) for the given AEIO_InSpecH. <pre>AEGP_GetInSpecOptionsHandle(AEIO_InSpecH inH, void **optionsPPV);</pre>
AEGP_SetInSpecOptionsHandle	Sets the options data for the given AEIO_InSpecH. Must be allocated using the MemorySuite . <pre>AEGP_SetInSpecOptionsHandle(AEIO_InSpecH inH, void *optionsPV, void **old_optionsPPV);</pre>
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. <pre>AEGP_GetInSpecFilePath(AEIO_InSpecH inH, AEGP_MemHandle *file_nameZ);</pre>
AEGP_GetInSpecNativeFPS	Retrieves the frame rate of the AEIO_InSpecH. <pre>AEGP_GetInSpecNativeFPS(AEIO_InSpecH inH, A_Fixed *native_fpsP);</pre>
AEGP_SetInSpecNativeFPS	Sets the frame rate of the AEIO_InSpecH. <pre>AEGP_SetInSpecNativeFPS(AEIO_InSpecH inH, A_Fixed native_fpsP);</pre>
AEGP_GetInSpecDepth	Retrieves the bit depth of the image data in the AEIO_InSpecH. <pre>AEGP_GetInSpecDepth(AEIO_InSpecH inH, A_short *depthPS);</pre>

TABLE 110: AEGPIOINSUITE4

Function	Purpose
AEGP_SetInSpecDepth	Indicates to After Effects the bit depth of the image data in the AEIO_InSpecH. <pre>AEGP_SetInSpecDepth(AEIO_InSpecH inH, A_short depthS);</pre>
AEGP_GetInSpecSize	Retrieves the size (in bytes) of the data referenced by the AEIO_InSpecH. <pre>AEGP_GetInSpecLargeSize(AEIO_InSpecH inH, AEIO_FileSize *sizePLLu);</pre>
AEGP_SetInSpecSize	Indicates to After Effects the size (in bytes) of the data referenced by the AEIO_InSpecH. <pre>AEGP_SetInSpecLargeSize(AEIO_InSpecH inH, AEIO_FileSize sizeL);</pre>
AEGP_GetInSpecInterlaceLabel	Retrieves field information for the AEIO_InSpecH. <pre>AEGP_GetInSpecInterlaceLabel(AEIO_InSpecH inH, FIEL_Label *interlaceP);</pre>
AEGP_SetInSpecInterlaceLabel	Specifies field information for the AEIO_InSpecH. <pre>AEGP_SetInSpecInterlaceLabel(AEIO_InSpecH inH, const FIEL_Label *interlaceP);</pre>
AEGP_GetInSpecAlphaLabel	Retrieves alpha channel interpretation information for the AEIO_InSpecH. <pre>AEGP_GetInSpecAlphaLabel(AEIO_InSpecH inH, AEIO_AlphaLabel *alphaP);</pre>
AEGP_SetInSpecAlphaLabel	Sets alpha channel interpretation information for the AEIO_InSpecH. <pre>AEGP_SetInSpecAlphaLabel(AEIO_InSpecH inH, const AEIO_AlphaLabel *alphaP);</pre>
AEGP_GetInSpecDuration	Retrieves the duration of the AEIO_InSpecH. <pre>AEGP_GetInSpecDuration(AEIO_InSpecH inH, A_Time *durationP);</pre>

TABLE 110: AEGPIOINSUITE4

Function	Purpose
<code>AEGP_SetInSpecDuration</code>	<p>Sets the duration of the <code>AEIO_InSpecH</code>. NOTE: As of 5.5, this must be called, even for frame-based file formats. If you don't set the <code>A_Time.scale</code> to something other than zero, your file(s) will not import. This will be fixed in future versions.</p> <pre>AEGP_SetInSpecDuration(AEIO_InSpecH inH, const A_Time *durationP);</pre>
<code>AEGP_GetInSpecDimensions</code>	<p>Retrieves the width and height of the image data in the <code>AEIO_InSpecH</code>.</p> <pre>AEGP_GetInSpecDimensions(AEIO_InSpecH inH, A_long *widthPL0, A_long *heightPL0);</pre>
<code>AEGP_SetInSpecDimensions</code>	<p>Indicates to After Effects the width and height of the image data in the <code>AEIO_InSpecH</code>.</p> <pre>AEGP_SetInSpecDimensions(AEIO_InSpecH inH, A_long widthL, A_long heightL);</pre>
<code>AEGP_AddAuxExtMap</code>	<p>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.</p> <pre>AEGP_AddAuxExtMap(const A_char *extension, A_long file_type, A_long creator);</pre>
<code>AEGP_InSpecGetRationalDimension</code>	<p>Retrieves the width, height, bounding rect, and scaling factor applied to an <code>AEIO_InSpecH</code>.</p> <pre>AEGP_InSpecGetRationalDimensions(AEIO_InSpecH inH, const AEIO_RationalScale *rs0, A_long *width0, A_long *height0, A_Rect *r0);</pre>
<code>AEGP_GetInSpecHSF</code>	<p>Retrieves the horizontal scaling factor applied to an <code>AEIO_InSpecH</code>.</p> <pre>AEGP_GetInSpecHSF(AEIO_InSpecH inH, A_Ratio *hsf);</pre>

TABLE 110: AEGPIOINSUITE4

Function	Purpose
<code>AEGP_SetInSpecHSF</code>	Sets the horizontal scaling factor of an <code>AEIO_InSpecH</code> . <pre>AEGP_SetInSpecHSF(AEIO_InSpecH inH, const A_Ratio *hsf);</pre>
<code>AEGP_GetInSpecSoundRate</code>	Obtains the sampling rate (in samples per second) for the audio data referenced by the <code>AEIO_InSpecH</code> . <pre>AEGP_GetInSpecSoundRate(AEIO_InSpecH inH, A_FpLong *ratePF);</pre>
<code>AEGP_SetInSpecSoundRate</code>	Sets the sampling rate (in samples per second) for the audio data referenced by the <code>AEIO_InSpecH</code> . <pre>AEGP_SetInSpecSoundRate(AEIO_InSpecH inH, A_FpLong rateF);</pre>
<code>AEGP_GetInSpecSoundEncoding</code>	Obtains the encoding method (signed PCM, unsigned PCM, or floating point) from an <code>AEIO_InSpecH</code> . <pre>AEGP_GetInSpecSoundEncoding(AEIO_InSpecH inH, AEIO_SndEncoding *encodingP);</pre>
<code>AEGP_SetInSpecSoundEncoding</code>	Sets the encoding method of an <code>AEIO_InSpecH</code> . <pre>AEGP_SetInSpecSoundEncoding(AEIO_InSpecH inH, AEIO_SndEncoding encoding);</pre>
<code>AEGP_GetInSpecSoundSampleSize</code>	Retrieves the bytes-per-sample (1,2, or 4) from an <code>AEIO_InSpecH</code> . <pre>AEGP_GetInSpecSoundSampleSize(AEIO_InSpecH inH, AEIO_SndSampleSize *bytes_per_smpP);</pre>
<code>AEGP_SetInSpecSoundSampleSize</code>	Set the bytes per sample of an <code>AEIO_InSpecH</code> . <pre>AEGP_SetInSpecSoundSampleSize(AEIO_InSpecH inH, AEIO_SndSampleSize bytes_per_sample);</pre>
<code>AEGP_GetInSpecSoundChannels</code>	Determines whether the audio in the <code>AEIO_SndChannels</code> is mono or stereo. <pre>AEGP_GetInSpecSoundChannels(AEIO_InSpecH inH, AEIO_SndChannels *num_channelsP);</pre>

TABLE 110: AEGPIOINSUITE4

Function	Purpose
<code>AEGP_SetInSpecSoundChannels</code>	Sets the audio in an <code>AEIO_SndChannels</code> to mono or stereo. <pre>AEGP_SetInSpecSoundChannels(AEIO_InSpecH inH, AEIO_SndChannels num_channels);</pre>
<code>AEGP_SetInSpecCMOptions</code>	Specifies desired color management behavior for the given <code>AEIO_InSpecH</code> . If <code>force_to_pwsB</code> is <code>TRUE</code> , the <code>AEIO_InSpecH</code> will be in the working colorspace even if it has an embedded profile. Pass <code>TRUE</code> for <code>disable_cm_uiB</code> to disable the color management UI. <pre>AEGP_SetInSpecCMOptions(AEIO_InSpecH inH, A_Boolean force_to_pwsB, A_Boolean disable_cm_uiB);</pre>

WHAT GOES OUT

These functions manage all interactions with an output specification in After Effects' render queue.

TABLE 111: AEGPIOOUTSUITE4

Function	Purpose
<code>AEGP_GetOutSpecOptionsHandle</code>	Retrieves the Options for the <code>AEIO_OutSpecH</code> . <pre>AEGP_GetOutSpecOptionsHandle(AEIO_OutSpecH outH, void **optionsPPV);</pre>
<code>AEGP_SetOutSpecOptionsHandle</code>	Sets the Options for the <code>AEIO_OutSpecH</code> . <pre>AEGP_SetOutSpecOptionsHandle(AEIO_OutSpecH outH, void *optionsPV, void **old_optionsPPV);</pre>

TABLE 111: AEGPIOOUTSUITE4

Function	Purpose
<code>AEGP_GetOutSpecFilePath</code>	<p>Obtains the path for the <code>AEIO_OutSpecH</code>. The file path is a handle to a NULL-terminated <code>A_UTF16Char</code> string, and must be disposed with <code>AEGP_FreeMemHandle</code>.</p> <p>If <code>file_rsrvdPB</code> returns <code>TRUE</code>, the plug-in should not overwrite it (After Effects has already created an empty file); doing so can cause network renders to fail.</p> <pre>AEGP_GetOutSpecFilePath(AEIO_OutSpecH outh, AEGP_MemHandle *unicode_pathPH, A_Boolean *file_rsrvdPB);</pre>
<code>AEGP_GetOutSpecFPS</code>	<p>Obtains the frames per second of the <code>AEIO_OutSpecH</code>.</p> <pre>AEGP_GetOutSpecFPS(AEIO_OutSpecH outh, A_Fixed *native_fpsP);</pre>
<code>AEGP_SetOutSpecNativeFPS</code>	<p>Sets the frames per second of the <code>AEIO_OutSpecH</code>.</p> <pre>AEGP_SetOutSpecNativeFPS(AEIO_OutSpecH outh, A_Fixed native_fpsP);</pre>
<code>AEGP_GetOutSpecDepth</code>	<p>Obtains the pixel bit depth of the <code>AEIO_OutSpecH</code>.</p> <pre>AEGP_GetOutSpecDepth(AEIO_OutSpecH outh, A_short *depthPS);</pre>
<code>AEGP_SetOutSpecDepth</code>	<p>Sets the pixel bit depth of the <code>AEIO_OutSpecH</code>.</p> <pre>AEGP_SetOutSpecDepth(AEIO_OutSpecH outh, A_short depthPS);</pre>
<code>AEGP_GetOutSpecInterlaceLabel</code>	<p>Obtains field information for the <code>AEIO_OutSpecH</code>.</p> <pre>AEGP_GetOutSpecInterlaceLabel(AEIO_OutSpecH outh, FIEL_Label *interlaceP);</pre>
<code>AEGP_SetOutSpecInterlaceLabel</code>	<p>Set the field information for the <code>AEIO_OutSpecH</code>.</p> <pre>AEGP_SetOutSpecInterlaceLabel(AEIO_OutSpecH outh, const FIEL_Label *interlaceP);</pre>

TABLE 111: AEGPIOOUTSUITE4

Function	Purpose
AEGP_GetOutSpecAlphaLabel	Obtains alpha interpretation information for the AEIO_OutSpecH. <pre>AEGP_GetOutSpecAlphaLabel(AEIO_OutSpecH outh, AEIO_AlphaLabel *alphaP);</pre>
AEGP_SetOutSpecAlphaLabel	Sets the alpha interpretation for the AEIO_OutSpecH. <pre>AEGP_SetOutSpecAlphaLabel(AEIO_OutSpecH outh, const AEIO_AlphaLabel *alphaP);</pre>
AEGP_GetOutSpecDuration	Obtains the duration of the AEIO_OutSpecH. <pre>AEGP_GetOutSpecDuration(AEIO_OutSpecH outh, A_Time *durationP);</pre>
AEGP_SetOutSpecDuration	Sets the duration of the AEIO_OutSpecH. <pre>AEGP_SetOutSpecDuration(AEIO_OutSpecH outh, const A_Time *durationP);</pre>
AEGP_GetOutSpecDimensions	Obtains the dimensions of the AEIO_OutSpecH. <pre>AEGP_GetOutSpecDimensions(AEIO_OutSpecH outh, A_long *widthPL0, A_long *heightPL0);</pre>
AEGP_GetOutSpecHSF	Obtains the horizontal scaling factor of the AEIO_OutSpecH. <pre>AEGP_GetOutSpecHSF(AEIO_OutSpecH outh, A_Ratio *hsf);</pre>
AEGP_SetOutSpecHSF	Sets the horizontal scaling factor of the AEIO_OutSpecH. <pre>AEGP_SetOutSpecHSF(AEIO_OutSpecH outh, const A_Ratio *hsf);</pre>
AEGP_GetOutSpecSoundRate	Obtains the sampling rate for the AEIO_OutSpecH. <pre>AEGP_GetOutSpecSoundRate(AEIO_OutSpecH outh, A_FpLong *ratePF);</pre>

TABLE 111: AEGPIOOUTSUITE4

Function	Purpose
<code>AEGP_SetOutSpecSoundRate</code>	Sets the sampling rate for the <code>AEIO_OutSpecH</code> . <pre>AEGP_SetOutSpecSoundRate(AEIO_OutSpecH outh, A_FpLong rateF);</pre>
<code>AEGP_GetOutSpecSoundEncoding</code>	Obtains the sound encoding format of the <code>AEIO_OutSpecH</code> . <pre>AEGP_GetOutSpecSoundEncoding(AEIO_OutSpecH outh, AEIO_SndEncoding *encodingP);</pre>
<code>AEGP_SetOutSpecSoundEncoding</code>	Sets the sound encoding format of the <code>AEIO_OutSpecH</code> . <pre>AEGP_SetOutSpecSoundEncoding(AEIO_OutSpecH outh, AEIO_SndEncoding encoding);</pre>
<code>AEGP_GetOutSpecSoundSampleSize</code>	Obtains the bytes-per-sample of the <code>AEIO_OutSpecH</code> . <pre>AEGP_GetOutSpecSoundSampleSize(AEIO_OutSpecH outh, AEIO_SndSampleSize *bpsP);</pre>
<code>AEGP_SetOutSpecSoundSampleSize</code>	Sets the bytes-per-sample of the <code>AEIO_OutSpecH</code> . <pre>AEGP_SetOutSpecSoundSampleSize(AEIO_OutSpecH outh, AEIO_SndSampleSize bpsP);</pre>
<code>AEGP_GetOutSpecSoundChannels</code>	Obtains the number of sounds channels in the <code>AEIO_OutSpecH</code> . <pre>AEGP_GetOutSpecSoundChannels(AEIO_OutSpecH outh, AEIO_SndChannels *channelsP);</pre>
<code>AEGP_SetOutSpecSoundChannels</code>	Sets the number of sounds channels in the <code>AEIO_OutSpecH</code> . <pre>AEGP_SetOutSpecSoundChannels(AEIO_OutSpecH outh, AEIO_SndChannels channels);</pre>
<code>AEGP_GetOutSpecIsStill</code>	Determines whether the <code>AEIO_OutSpecH</code> is a still. <pre>AEGP_GetOutSpecIsStill(AEIO_OutSpecH outh, A_Boolean *is_stillPB);</pre>

TABLE 111: AEGPIOOUTSUITE4

Function	Purpose
AEGP_GetOutSpecPosterTime	Obtains the time of the AEIO_OutSpecH's poster frame. <pre>AEGP_GetOutSpecPosterTime(AEIO_OutSpecH outh, A_Time *poster_timeP);</pre>
AEGP_GetOutSpecStartFrame	Obtains the time of the first frame in the AEIO_OutSpecH. <pre>AEGP_GetOutSpecStartFrame(AEIO_OutSpecH outh, A_long *start_frameP);</pre>
AEGP_GetOutSpecPullDown	Obtains the pulldown phase of the AEIO_OutSpecH. <pre>AEGP_GetOutSpecPullDown(AEIO_OutSpecH outh, AEIO_Pulldown *pulldownP);</pre>
AEGP_GetOutSpecIsMissing	Sets the pulldown phase of the AEIO_OutSpecH. <pre>AEGP_GetOutSpecIsMissing(AEIO_OutSpecH outh, A_Boolean *missingPB);</pre>
AEGP_GetOutSpecShouldEmbedICCProfile	Returns TRUE if the AEIO should embed a color profile in the output. <pre>AEGP_GetOutSpecShouldEmbedICCProfile(AEIO_OutSpecHouth, A_Boolean *embedPB);</pre>
AEGP_GetNewOutSpecICCProfile	Returns an (opaque) ICC color profile for embedding in the AEIO's output. Must be disposed by caller. <pre>AEGP_GetNewOutSpecICCProfile(AEGP_PluginID aegp_plugin_id, AEIO_OutSpecHouth, AEGP_MemHandle *icc_profilePH);</pre>
AEGP_GetOutSpecRQItem	Returns the AEGP_RQItemRefH associated with the given AEIO_OutSpecH. Important for using the AEGP_RQItemSuite . <pre>AEGP_GetOutSpecRQItem(AEIO_OutSpecHouth, AEGP_RQItemRefH *rq_itemP);</pre>

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, et cetera. We use option handles for information about the file; output settings, dimensions, details of compression settings used.

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\(\)](#).

When users select a file in the file import dialog which is of a type handled by your AEIO, its [AEIO_VerifyFileImportable\(\)](#) function will be called; it's called again for each such file the user imports. [AEIO_InitInSpecFromFile\(\)](#) 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\(\)](#). After Effects then calls the plug-in's [AEIO_GetInSpecInfo\(\)](#) 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\(\)](#) is then sent; if the AEIO indicates that there is user data present, [AEIO_GetUserData\(\)](#) 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\(\)](#).

Once the supported file is added to a composition, user interaction will generate calls to [AEIO_DrawSparseFrame\(\)](#) and [AEIO_GetSound\(\)](#).

When the project is saved, and if there is options data associated with the AEIO_InSpec, After Effects will send [AEIO_FlattenOptions\(\)](#) 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](#) being sent.

If the user adds an item to the render queue and chooses the AEIO's supported output format, [AEIO_InitOutputSpec\(\)](#) will be sent. Use the various get functions to obtain information about the output settings, and store any pertinent information using [AEIO_SetOutSpecOptionsHandle\(\)](#), followed by [AEIO_GetFlatOutputOptions\(\)](#). [AEIO_GetDepths\(\)](#) is sent so After Effects can determine what output pixel bit depths the AEIO supports.

[AEIO_GetOutputInfo\(\)](#) is sent so that file name, type and subtype information can be displayed in the output module details.

[AEIO_UserOptionsDialog\(\)](#) is called when the user clicks on the Format Options button, in the render queue.

When the user actually clicks on the "Render" button, [AEIO_SetOutputFile](#) will be called, followed by [AEIO_GetSizes\(\)](#) (your AEIO is responsible for determining whether the file to be rendered will fit on the output volume).

If the AEIO supports a video or audio format, [AEIO_StartAdding\(\)](#) is sent to notify the AEIO that data is coming next. If the AEIO supports a frame-based format, [AEIO_OutputFrame\(\)](#) is called repeatedly. After Effects sends a `PF_EffectWorld` representation of the frame to be output. [AEIO_WriteLabels\(\)](#) is called (for each frame) to give the plug-in a chance to write out field and alpha interpretation information. [AEIO_EndAdding\(\)](#) is sent when there are no more frames (or audio) to be output. Close the output file.

10 • PREMIERE PRO & OTHER HOSTS

Adobe Premiere Pro and Adobe Premiere Elements both support the After Effects effect API. They offer a thorough host implementation, some the key omissions being 3D-related calls (auxiliary channel information, cameras and lights) 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 as the same support exists in corresponding versions of Premiere Elements.

Application Versions	PF_InData> version.major	PF_InData> version.minor
Premiere Pro CS5	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.

DIFFERENT PIXEL FORMATS

Premiere Pro provides function suites for enumerating supported pixel formats other than ARGB, such as YUV. Use the `PF_Pixel Format Suite` (defined in `PrAESDKSupport.h`) to register for [PF_EffectWorlds](#) in other pixel formats, and manage worlds in those other pixel formats. Use the `Premiere Pixel Format Suite` (defined in the aptly-named `PrSDKPixelFormatSuite.h`) to get black and white values, and convert between pixel formats.

32-BIT FLOATING POINT SUPPORT

Premiere Pro does not support SmartFX. Using a 32-bit pixel format with `PF_Cmd_RENDER` is the way to support 32-bit rendering in Premiere Pro. See the SDK Noise sample project for an example implementation.

As 32-bit processing can be *ahem* “challenging” to render in real-time, so there are various settings to enable and disable 32-bit rendering. Settings>Sequence Settings>Video Previews>Maximum Bit Depth controls previewing from the timeline. For export to file, use Export Settings>Video>Basic Settings>Render at Maximum Depth.

As of CS5, `PF_CHECKOUT_PARAM()` only supports returns 8-bit ARGB buffers, regardless of the pixel format currently being used for rendering. So if you rely heavily on this callback and wish to support 32-bit rendering, [let us know](#).

MULTITHREADING

Starting with Premiere Pro CS3, Premiere uses multithreaded rendering for AE effects. For AE effects, there is 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. Effects should not depend on static, non-constant variables.

New in Premiere Pro CS4.1, we have added a new flag to the AE API, `PF_OutFlag2_PPPO_DO_NOT_CLONE_SEQUENCE_DATA_FOR_RENDER`, to allow a plug-in to opt out of Premiere Pro's multithreaded rendering of AE effects. 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.

PARAMETERS

Premiere Pro does not honor the `PF_ParamFlag_START_COLLAPSED` flag. Parameters are always initialized with their twirlies collapsed, and cannot be automatically twirled open by parameter supervision.

TIME DIFFERENCES

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.

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`](#). This differs from After Effects, where the source layer's quality setting is provided here.

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. If your effect needs to be reloaded each time, there is a way to disable this caching. From the Premiere Pro SDK, the plug-in can use the PF Cache On Load Suite in `AE_CacheOnLoadSuite.h` to call `PF_SetNoCacheOnLoad()` during `PF_Cmd_GLOBAL_SETUP`.

PLUG-IN INSTALLATION

Use the common plug-in folder as described [here](#). 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. This is because Premiere Pro's export goes through Adobe Media Encoder, an entirely separate application. Oh, and you'll also miss out on copy and paste between Premiere Pro and After Effects.

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.

EFFECT THUMBNAIL PREVIEWS

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](#), prefixed with “AE.” So if the match name was “MatchName”, then the filename should be “AE.MatchName.png”

GENERAL GUIDELINES

We’ve tried to provide robust compatibility for After Effects effect plug-ins in Premiere Pro. We realize the API implementation may not be perfect; if you run into anomalous or just plain confusing behavior, please [let us know](#) as soon as possible. Speaking of anomalous behavior...

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](#) with the feature request. Numbers preceded by an ‘F’ are feature request numbers, and the others are bug numbers)

- F7233 - extent_hint support
- F7292 - PF_AppGetGetColor should return background color of application
- 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
- F9360 - Support PF_OutFlag2_PARAM_GROUP_START_COLLAPSED_FLAG and PF_ParamFlag_START_COLLAPSED, so that parameters can be initialized in an expanded state
- PF World Transform Suite
- PF AE Channel Suite
- AE’s implementation of high bit color depth support
- SmartFX
- 3D support
- PF_BLEND(), PF_SUBPIXEL_SAMPLE(), PF_GET_PIXEL_DATA16()
- 1235407 – No custom ECW UI over a standard data type

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
Coding conventions	5
Sample Project Descriptions.....	30
Effect API Versions.....	35
Effect plug-in entry point	39
Command Selectors.....	40
PF_InData.....	46
PF_OutData.....	51
PF_OutFlags.....	52
PF_OutFlags2.....	56
Parameter Types.....	59
PF_ParamDef.....	62
Parameter UI Flags	63
Parameter Flags	64
PF_EffectWorld structure	65
PF_PixelPtr accessor macros	68
Error Codes.....	69
PF_HandleSuite1	73
PF_WorldSuite1	73
Iteration suite functions.....	75
PF_WorldTransformSuite1	78
Kernel Flags.....	81
PF_FillMatteSuite2.....	82
PF_SamplingSuite Functions (multiple suites).....	83
PF_BatchSamplingSuite1.....	84
PF_ANSICallbacksSuite1	85
Interaction Callbacks	87
PF_ParamUtilsSuite1	95
Arbitrary data selectors	100
PF_EffectUISuite.....	103
PF_AppSuite4	104
AE_AdvAppSuite2	106
PF_AdvTimeSuite1.....	107
PF_AdvItemSuite1	108
PF_ChannelSuite1.....	109
PF_ColorParamSuite1	111
PF_PathVertex.....	112
PF_PathDataSuite1.....	113
PF_PathQuerySuite	115
Pixel Types for different color spaces	117
color space conversion callbacks.....	117
PF_PreRenderExtra	127
PF_PreRenderOutput.....	129
PF_SmartRenderExtra	132

Events.....	135
PF_EventExtra.....	136
PF_Context.....	138
PF_EffectWindowInfo.....	138
PF_DoClickEventInfo.....	139
PF_DrawEventInfo.....	139
PF_KeyDownEvent.....	140
PF_AdjustCursorEventInfo.....	142
PF_ArbParamsExtra.....	142
PF_EffectCustomUISuite1.....	144
DRAWBOT_DrawbotSuite1.....	144
DRAWBOT_SupplierSuite1.....	145
DRAWBOT_SurfaceSuite1.....	147
DRAWBOT_PathSuite1.....	150
PF_EffectCustomUIOverlayThemeSuite1.....	151
UI Callbacks.....	154
Audio data structures.....	157
AEGP API Data Types.....	162
Data types requiring disposal.....	169
AEGP Suites.....	170
AEGP_FIMSuite3.....	172
AEGP_ProjSuite5.....	173
AEGP_TimeDisplay2.....	175
AEGP_ItemSuite8.....	176
AEGP_ColorSettingsSuite2.....	181
AEGP_CompSuite7.....	183
AEGP_FootageSuite5.....	188
AEGP_FootageInterp structure.....	194
AEGP_SoundDataSuite1.....	196
AEGP_LayerSuite7.....	197
AEGP_StreamSuite4.....	205
AEGP_MarkerSuite2.....	212
AEGP_DynamicStreamSuite4.....	215
AEGP_KeyframeSuite3.....	222
AEGP_TextDocumentSuite1.....	227
AEGP_TextLayerSuite1.....	228
AEGP_RenderSuite2.....	229
AEGP_EffectSuite3.....	231
AEGP_RegisterSuite5.....	235
AEGP_CommandSuite1.....	237
AEGP_UtilitySuite5.....	239
AEGP_MemorySuite1.....	244
AEGP_MaskSuite5.....	245
AEGP_MaskOutlineSuite2.....	248
AEGP_PersistentDataSuite3.....	250

AEGP_PFIInterfaceSuite1	254
AEGP_RenderQueueSuite1	255
AEGP_RQItemSuite3	256
AEGP_RenderOptionsSuite3	258
AEGP_OutputModuleSuite4	262
AEGP_CollectionSuite2	266
AEGP_WorldSuite3	267
AEGP_IterateSuite1	270
Data types used in the Artisan API	274
Artisan Entry Points	275
AEGP_RenderSuite	278
AEGP_CanvasSuite7	279
AEGP_ArtisanUtilSuite1	288
AEGP_CompositeSuite2	289
AEGP_CameraSuite2	292
AEGP_LightSuite2	294
AEGP_QueryXformSuite2	295
PR_InteractiveDrawProcs	298
AEIO_ModuleInfo	301
AEIO_ModuleFlags	302
AEIO_ModuleFlags2	304
AEIO_FunctionBlock4	305
AEGPIOInSuite4	316
AEGPIOOutSuite4	320

A

AEFX_CLR_STRUCT	59
AEGP_StreamSuite1	205
area	138
Audio-specific Float Slider Variables	158

C

cbs	137
cmd	39
continue_refcon	139
current_frame	138

D

data	65, 66
Debugging	159
depth	139
Draw Event	139

E

effect_win	137
evt_in_flags	137
evt_out_flags	137
extent_hint	66

F

flat_sdata_size	51
frame_data	51

G

GET_LAYER2COMP_XFORM	155
global_data	51
Graphics Utility Callbacks	85

H

height	51, 66
HLS pixel	117

horiz_offset 138

I

in_data 39

index 138

K

Key Down Event 140

L

last_time 139

luminance 118

M

modifiers 139

my_version 51

N

name 51

num_clicks 139

num_params 51

O

origin 51

out_data 39

out_flags 51

P

param_title_frame 138

Parameters 51

PF_ABORT 87

PF_ADD_PARAM 58

PF_Arbitrary_FLAT_SIZE_FUNC 100

PF_Arbitrary_INTERP_FUNC 101

PF_Arbitrary_PRINT_SIZE_FUNC 101

PF_CHECKIN_LAYER_AUDIO 89

PF_CHECKIN_PARAM 88

PF_CHECKOUT_LAYER_AUDIO 89

PF_CHECKOUT_PARAM	88, 123
PF_Cmd_ABOUT	39
PF_Cmd_GLOBAL_SETUP	53
PF_Cmd_PARAMS_SETUP	58
PF_DrawEventInfo	139
PF_EA_CONTROL	138
PF_EA_PARAM_TITLE	138
PF_Event_ACTIVATE	135
PF_Event_ADJUST_CURSOR	136
PF_Event_CLOSE_CONTEXT	136
PF_Event_DEACTIVATE	136
PF_Event_DO_CLICK	135
PF_Event_DRAG	136
PF_Event_DRAW	136
PF_Event_IDLE	136
PF_Event_KEYDOWN	136
PF_Event_NEW_CONTEXT	135
PF_ExtDependenciesExtra	44
PF_GET_AUDIO_DATA	89
PF_HLS_PIXEL	117
PF_InData	87
PF_InData Structure	46
PF_OutData structure	51
PF_Param_ANGLE	59
PF_Param_CHECKBOX	59
PF_Param_COLOR	59
PF_Param_FIX_SLIDER	59
PF_Param_POPUP	60
PF_Param_SLIDER	59
PF_ParamDef	88
PF_Pixel	117
PF_PlatformData_RES_FILE_PATH	155
PF_PROGRESS	87
PF_REGISTER_UI	88
PF_YIO_PIXEL	117
platform_ref	66

R

return_msg	51, 53
rowbytes	66

S

screen_point	139
--------------------	-----

send_drag 139

U

update_rect 139

W

when 139

width 51, 66

world_flags 65