ADOBE® AFTER EFFECTS® CS6 SCRIPTING GUIDE

Overview

Introduction to scripting in After Effects

A script is a series of commands that tells an application to perform a series of operations. You can use scripts in most Adobe applications to automate repetitive tasks, perform complex calculations, and even use some functionality not directly exposed through the graphical user interface. For example, you can direct After Effects to reorder the layers in a composition, find and replace source text in text layers, or send an e-mail message when rendering is complete.

See “Examples” on page 191 for examples of what scripts can do.

Although both the After Effects expressions language and the After Effects ExtendScript scripting language are based on JavaScript, the expressions features and scripting features of After Effects are separate and distinct. Expressions cannot access information from scripts (such as variables and functions). Whereas a script tells an application to *do* something, an expression says that a property *is* something. However, because the After Effects expression language and ExtendScript are both based on JavaScript, familiarity with either one is very helpful in understanding the other.

The heart of a scriptable application is the object model. When you use Adobe After Effects, you create projects, compositions, and render queue items along with all of the elements that they contain: footage, images, solids, layers, masks, effects, and properties. Each of these items, in scripting terms, is an *object*. This guide describes the ExtendScript objects that have been defined for After Effects projects.

The After Effects object model is composed of a project, items, compositions, layers, and render queue items. Each object has its own special attributes, and every object in an After Effects project has its own identity (although not all are accessible to scripting). You should be familiar with the After Effects object model in order to create scripts.

NOTE: JavaScript objects normally referred to as “properties” are consistently called “attributes” in this guide, to avoid confusion with After Effects’ own definition of a property (an animatable value of an effect, mask, or transform within an individual layer).

Nearly all of what scripting can accomplish replicates what can be done by means of the After Effects graphical user interface. A thorough knowledge of the application itself and its graphical user interface is essential to understanding how to use scripting in After Effects.

The ExtendScript language

After Effects scripts use the Adobe ExtendScript language, which is an extended form of JavaScript used by several Adobe applications, including Photoshop, Illustrator, and InDesign. ExtendScript implements the JavaScript language according to the ECMA-262 specification. The After Effects scripting engine supports the 3rd Edition of the ECMA-262 Standard, including its notational and lexical conventions, types, objects, expressions, and statements. ExtendScript also implements the E4X ECMA-357 specification, which defines access to data in XML format.

ExtendScript defines a global debugging object, the dollar ($) object, and a reporting utility for ExtendScript elements, the ExtendScript Reflection interface.

File and Folder Objects: Because path name syntax is very different in different operating systems, Adobe ExtendScript defines `File` and `Folder` objects to provide platform-independent access to the underlying file system.

ScriptUI User Interface Module: The ExtendScript ScriptUI module provides the ability to create and interact with user interface elements. ScriptUI provides an object model for windows and UI control elements that you can use to create a user interface for your scripts.

Tools and Utilities: In addition, ExtendScript provides tools and features such as a localization utility for providing user-interface string values in different languages and global functions for displaying short messages in dialog boxes (`alert`, `confirm`, and `prompt`).

External Communication: ExtendScript provides a `Socket` object that allows you to communicate with remote systems from your After Effects scripts.

Interapplication Communication: ExtendScript provides a common scripting environment for all Adobe applications, and allows interapplication communication through scripts.

The ExtendScript Toolkit (ESTK)

After Effects includes a script editor and debugger, the ExtendScript Toolkit (ESTK), which provides a convenient interface for creating and testing your own scripts.

To start the ESTK, choose File > Scripts > Open Script Editor.

If you choose to use another text editor to create, edit, and save scripts, be sure to choose an application that does not automatically add header information when saving files and that saves with Unicode (UTF-8) encoding. In many text editors, you can set preferences for saving with UTF-8 encoding. Some applications (such as Microsoft Word) by default add header information to files that can cause “line 0” errors in scripts, causing them to fail.

For detailed information on the ExtendScript Toolkit, see the *JavaScript Tools Guide*.

The .jsx and .jsxbin file-name extensions

ExtendScript script files are distinguished by the `.jsx` file-name extension, a variation on the standard `.js` extension used with JavaScript files. After Effects scripts must include the `.jsx` file extension in order to be properly recognized by the application. Any UTF-8-encoded text file with the `.jsx` extension is recognized as an ExtendScript file.

You can use the ExtendScript Toolkit to export a binary version of an ExtendScript file, which has the extension `.jsxbin`. Such a binary file may not be usable with all of the scripting integration features in After Effects.

Activating full scripting features

The default is for scripts to not be allowed to write files or send or receive communication over a network. To allow scripts to write files and communicate over a network, choose Edit > Preferences > General (Windows) or After Effects > Preferences > General (Mac OS), and select the Allow Scripts To Write Files And Access Network option.

Any After Effects script that contains an error preventing it from being completed generates an error message from the application. This error message includes information about the nature of the error and the line of the script on which it occurred. The ExtendScript Toolkit (ESTK) debugger can open automatically when the application encounters a script error. This feature is disabled by default so that casual users do not encounter it. To activate this feature, choose Preferences > General, and select Enable JavaScript Debugger.

Loading and running scripts

Running scripts directly from the File > Scripts menu

When After Effects starts, it searches the Scripts folder for scripts to load. Loaded scripts are available from the File > Scripts menu.

To run a loaded script, choose File > Scripts > [script name].

If you edit a script while After Effects is running, you must save your changes for the changes to be applied. If you place a script in the Scripts folder while After Effects is running, you must restart After Effects for the script to appear in the Scripts menu, though you can immediately run the new script using the Run Script File command.

Running scripts using File > Scripts > Run Script File

To run a script that has not been loaded, choose File > Scripts > Run Script File, locate and select a script, and click Open.

Running scripts from the command line, a batch file, or an AppleScript script

If you are familiar with how to run a script from the command line in Windows or via AppleScript, you can send a script directly to the open After Effects application, so that the application automatically runs the script.

To run a script from the command line, call afterfx.exe from the command line. Use the -r switch and the full path of the script to run as arguments. This command does not open a new instance of the After Effects application; it runs the script in the existing instance.

Example (for Windows):

```
afterfx -r c:\script_path\example_script.jsx
```

You can use this command-line technique—together with the software that comes with a customizable keyboard—to bind the invocation of a script to a keyboard shortcut.

Following are examples of Windows command-line entries that will send an After Effects script to the application without using the After Effects user interface to execute the script.

In the first example, you copy and paste your After Effects script directly on the command line and then run it. The script text appears in quotation marks following the `afterfx.exe -s` command:

```
afterfx.exe -s "alert("You just sent an alert to After Effects")"
```

Alternatively, you can specify the location of the JSX file to be executed. For example:

```
afterfx.exe -r c:\myDocuments\Scripts\yourAEScriptHere.jsx
afterfx.exe -r "c:\myDocuments\Scripts\Script Name with Spaces.jsx"
```

How to include After Effects scripting in an AppleScript (Mac OS)

Following are three examples of AppleScript scripts that will send an existing JSX file containing an After Effects script to the application without using the After Effects user interface to execute the script.

In the first example, you copy your After Effects script directly into the Script Editor and then run it. The script text appears within quotation marks following the DoScript command, so internal quotes in the script must be escaped using the backslash escape character, as follows:

```
tell application "Adobe After Effects CS6"
  DoScript "alert(\"You just sent an alert to After Effects\")"
end tell
```

Alternatively, you could display a dialog box asking for the location of the JSX file to be executed, as follows:

```
set theFile to choose file
tell application "Adobe After Effects CS6"
  DoScript theFile
end tell
```

Finally, this script is perhaps most useful when you are working directly on editing a JSX script and want to send it to After Effects for testing or to run. To use it effectively you must enter the application that contains the open JSX file (in this example it is TextEdit); if you do not know the proper name of the application, type in your best guess to replace “TextEdit” and AppleScript prompts you to locate it.

Simply highlight the script text that you want to run, and then activate this AppleScript:

```
(*
This script sends the current selection to After Effects as a script.
*)

tell application "TextEdit"
  set the_script to text of front document
end tell

tell application "Adobe After Effects CS6"
  activate
  DoScript the_script
end tell
```

Running scripts automatically during application startup or shutdown

Within the Scripts folder are two folders called Startup and Shutdown. After Effects runs scripts in these folders automatically, in alphabetical order, on starting and quitting, respectively.

In the Startup folder you can place scripts that you wish to execute at startup of the application. They are executed after the application is initialized and all plug-ins are loaded.

Scripting shares a global environment, so any script executed at startup can define variables and functions that are available to all scripts. In all cases, variables and functions, once defined by running a script that contains them, persist in subsequent scripts during a given After Effects session. Once the application is quit, all such globally defined variables and functions are cleared. Be sure to give variables in scripts unique names, so that a script does not inadvertently reassign global variables intended to persist throughout a session.

Attributes can also be added to existing objects such as the Application object (see “Application object” on page 17) to extend the application for other scripts.

The Shutdown folder scripts are executed as the application quits. This occurs after the project is closed but before any other application shutdown occurs.

Running scripts from the Window menu

Scripts in the ScriptUI Panels folder are available from the bottom of the Window menu. If a script has been written to provide a user interface in a dockable panel, the script should be put in the ScriptUI folder. ScriptUI panels work much the same as the default panels in the After Effects user interface.

Instead of creating a Window object and adding controls to it, a ScriptUI Panels script uses the `this` object that represents the panel. For example, the following code adds a button to a panel:

```
var myPanel = this;
myPanel.add("button", [10, 10, 100, 30], "Tool #1");
```

If your script creates its user interface in a function, you cannot use `this` as it will refer to the function itself, not the panel. In this case, you should pass the `this` object as an argument to your function. For example:

```
function createUI(thisObj) {
  var myPanel = thisObj;
  myPanel.add("button", [10, 10, 100, 30], "Tool #1");
  return myPanel;
  }
var myToolsPanel = createUI(this);
```

You cannot use the File > Scripts > Run Script File menu command to run a script that refers to `this`. To make your script work with either a Window object (accessible from the File > Scripts menu) or a native panel (accessible from the Window menu), check whether `this` is a Panel object. For example:

```
function createUI(thisObj) {
  var myPanel = (thisObj instanceof Panel) ? thisObj : new Window("palette", "My Tools",
    [100, 100, 300, 300]);
  myPanel.add("button", [10, 10, 100, 30], "Tool #1");
  return myPanel;
  }
var myToolsPanel = createUI(this);
```

Stopping a running script

A script can be stopped by pressing Esc or Cmd+period (in Mac OS) when the After Effects or the script's user interface has focus. However, a script that is busy processing a lot of data might not be very responsive.

After Effects scripting reference

This chapter lists and describes JavaScript classes, objects, methods, attributes, and global functions defined by After Effects.

The After Effects scripting engine supports ExtendScript, Adobe's extended version of JavaScript, which implements the 3rd Edition of the ECMA-262 Standard, including its notational and lexical conventions, types, objects, expressions and statements. For a complete listing of the keywords and operators included with ECMAScript, refer to `ECMA-262.pdf`, available at www.ecma-international.org/publications/standards/Ecma-262.htm. For an overview of the most common keywords and statements available from ECMA-262, see "JavaScript keywords and statement syntax" on page 8.

Elements of basic JavaScript relevant to After Effects scripting

JavaScript variables

Scripting shares a global environment, so any script executed at startup can define variables and functions that are available to all scripts. In all cases, variables and functions, once defined by running a script that contains them, persist in subsequent scripts during a given After Effects session. Once the application is quit, all such globally defined variables and functions are cleared. Scripters should be careful about giving variables in scripts unique names, so that a script does not inadvertently reassign global variables intended to persist throughout a session.

JavaScript keywords and statement syntax

Although it is not possible to provide an exhaustive resource describing usage of JavaScript, the following tables provide an overview of keywords, statements, operators, precedence, and associativity.

The following table lists and describes all keywords and statements recognized by the After Effects scripting engine.

Table 1 Keywords and Statement Syntax

Keyword/Statement	Description
`break`	Standard JavaScript; exit the currently executing loop.
`continue`	Standard JavaScript; cease execution of the current loop iteration.
`case`	Label used in a `switch` statement.
`default`	Label used in a `switch` statement when a `case` label is not found.
`do...while`	Standard JavaScript construct. Similar to the `while` loop, except loop condition evaluation occurs at the end of the loop.
`false`	Literal representing the Boolean false value.
`for`	Standard JavaScript loop construct.

Keyword/Statement	Description
`for...in`	Standard JavaScript construct. Provides a way to easily loop through the properties of an object.
`function`	Used to define a function.
`if/if...else`	Standard JavaScript conditional constructs.
`new`	Standard JavaScript constructor statement.
`null`	Assigned to a variable, array element, or object property to indicate that it does not contain a legal value.
`return`	Standard JavaScript way of returning a value from a function or exiting a function.
`switch`	Standard JavaScript way of evaluating a JavaScript expression and attempting to match the expression's value to a `case` label.
`this`	Standard JavaScript method of indicating the current object.
`true`	Literal representing the Boolean true value.
`undefined`	Indicates that the variable, array element, or object property has not yet been assigned a value.
`var`	Standard JavaScript syntax used to declare a local variable.
`while`	Standard JavaScript construct. Similar to the `do...while` loop, except loop condition evaluation occurs at the beginning of the loop.
`with`	Standard JavaScript construct used to specify an object to use in subsequent statements.

JavaScript operators

The following tables list and describe all operators recognized by the After Effects scripting engine and show the precedence and associativity for all operators.

Table 2 Description of Operators

Operators	Description
`new`	Allocate object.
`delete`	Deallocate object.
`typeof`	Returns data type.
`void`	Returns undefined value.
`.`	Structure member.
`[]`	Array element.
`()`	Function call.
`++`	Pre- or post-increment.
`--`	Pre- or post-decrement.
`-`	Unary negation or subtraction.
`~`	Bitwise NOT.
`!`	Logical NOT.
`*`	Multiply.
`/`	Divide.

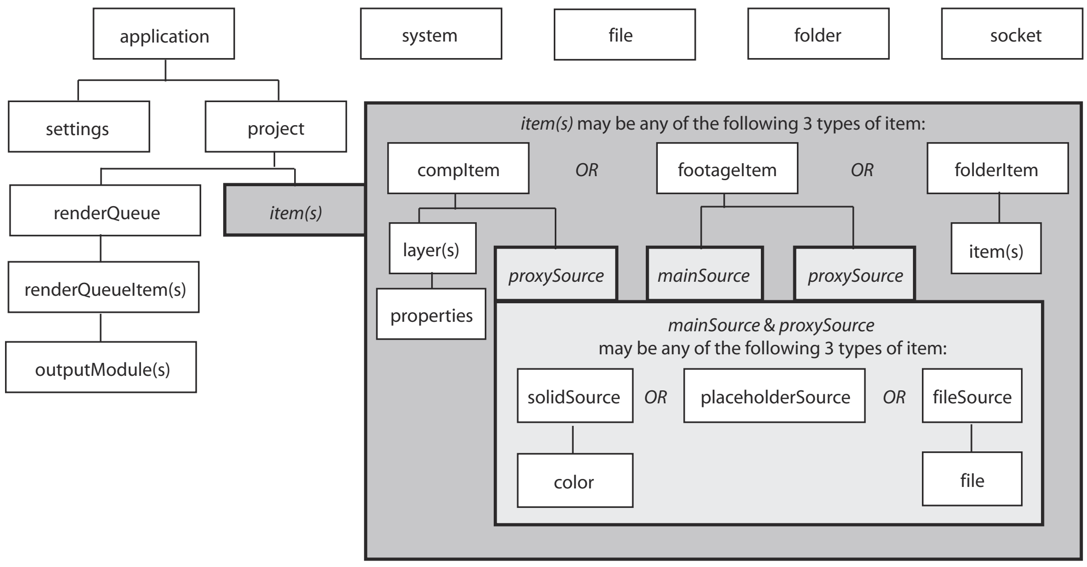
Operators	Description
%	Modulo division.
+	Add.
<<	Bitwise left shift.
>>	Bitwise right shift.
>>>	Unsigned bitwise right shift.
<	Less than.
<=	Less than or equal.
>	Greater than.
>=	Greater than or equal.
==	Equal.
!=	Not equal.
&	Bitwise AND.
^	Bitwise XOR.
|	Bitwise OR.
&&	Logical AND.
||	Logical OR.
?:	Conditional (ternary).
=	Assignment.
+=	Assignment with add operation.
-=	Assignment with subtract operation.
*=	Assignment with multiply operation.
/=	Assignment with divide operation.
%=	Assignment with modulo division operation.
<<=	Assignment with bitwise left shift operation.
>>=	Assignment with bitwise right shift operation.
>>>=	Assignment with unsigned bitwise right shift operation.
&=	Assignment with bitwise AND operation.
^=	Assignment with bitwise XOR operation.
|=	Assignment with bitwise OR operation.
,	Multiple evaluation.

Table 3 Operator Precedence

Operators (highest precedence to lowest)	Associativity
[], (), .	left to right
new, delete, – (unary negation), !, typeof, void, ++, ––	right to left
*, /, %	left to right
+, – (subtraction)	left to right
<<, >>, >>>	left to right
<, <=, >, >=	left to right
==, !=	left to right
&	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
=, /=, %=, <<=, >>=, >>>=, &=, ^=, |=, +=, –=, *=	right to left
,	left to right

The After Effects Object Model

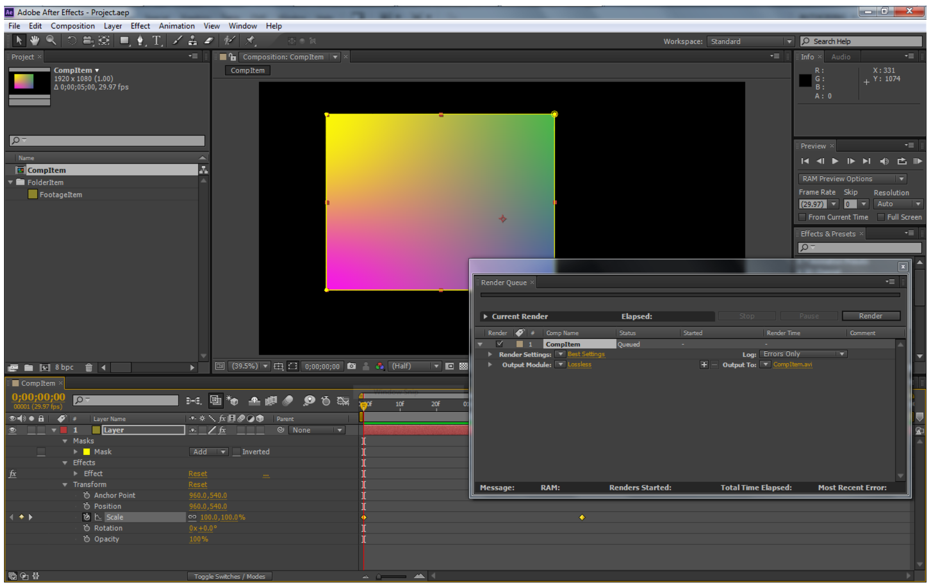
As you look through this reference section, which is organized alphabetically by object, you can refer to the following diagrams for an overview of where the various objects fall within the hierarchy, and their correspondence to the user interface.



Hierarchy diagram of the main After Effects scripting objects

Note that the File, Folder, and Socket objects are defined by ExtendScript, and are documented in the *JavaScript Tools Guide*. ExtendScript also defines the ScriptUI module, a set of window and user-interface control objects, which are available to After Effects scripts. These are also documented in the *JavaScript Tools Guide*.

The hierarchy of objects in scripting corresponds to the hierarchy in the user interface.



The application contains a Project panel, which displays a *project*. The project contains *compositions*, which contain *layers*. The source for a layer can be a *footage file*, *placeholder*, or *solid*, also listed in the Project panel. Each layer contains settings known as *properties*, and these can contain *markers* and *keyframes*. The *render queue* contains *render-queue items* as well as render settings and *output modules*. All of these entities are represented by objects in scripting.

NOTE: To avoid ambiguity, this manual uses the term "attribute" to refer to JavaScript object properties, and the term "property" or "AE property" to refer to After Effects layer properties.

Object summary

The following table lists all objects alphabetically, with links to the documentation page for each.

Object	**Description**
"Global functions" on page 14	Globally available functions that allow you to display text for script debugging purposes, and help convert time values between seconds and frames.
"Application object" on page 17	A single global object, available by its name (`app`), that provides access to objects and application settings within the After Effects application.
"AVItem object" on page 30	Represents audio/visual files imported into After Effects.
"AVLayer object" on page 38	Represents those layers that contain AVItem objects (composition layers, footage layers, solid layers, text layers, and sound layers).
"CameraLayer object" on page 50	Represents a camera layer within a composition.
"Collection object" on page 51	Associates a set of objects or values as a logical group and provides access to them by index.
"CompItem object" on page 52	Represents a composition, and allows you to manipulate it and get information about it.

Object	Description
“FileSource object” on page 61	Describes footage that comes from a file.
“FolderItem object” on page 63	Represents a folder in the Project panel.
“FootageItem object” on page 65	Represents a footage item imported into a project, which appears in the Project panel.
“FootageSource object” on page 69	Describes the file source of some footage.
“ImportOptions object” on page 75	Encapsulates options for importing files into After Effects.
“Item object” on page 78	Represents an item in a project that appears in the Project panel.
“ItemCollection object” on page 82	Collects items in a project.
“KeyframeEase object” on page 84	Encapsulates keyframe ease values in an After Effects property.
“Layer object” on page 86	A base class for layer classes.
“LayerCollection object” on page 95	Collects layers in a project.
“LightLayer object” on page 100	Represents a light layer within a composition.
“MarkerValue object” on page 102	Encapsulates marker values in an After Effects property.
“MaskPropertyGroup object” on page 106	Encapsulates mask attributes in a layer.
“OMCollection object” on page 109	Collects output modules in a render queue.
“OutputModule object” on page 110	Represents an output module for a render queue.
“PlaceholderSource object” on page 113	Describes a placeholder for footage.
“Project object” on page 114	Represents an After Effects project.
“Property object” on page 124	Represents an After Effects property.
“PropertyBase object” on page 148	A base class for After Effects property and property group classes.
“PropertyGroup object” on page 155	Represents an After Effects property group.
“RenderQueue object” on page 160	Represents the After Effects render queue.
“RenderQueueItem object” on page 163	Represents a renderable item in a render queue.
“RenderQueueItem object” on page 163	Collects render-queue items in a render queue.
“RQItemCollection object” on page 169	Provides access to application settings and preferences.
“Shape object” on page 172	Encapsulates the outline shape information for a mask.
“ShapeLayer object” on page 178	Represents a shape layer within a composition.
“SolidSource object” on page 179	Describes a solid color that is the source of some footage.
“System object” on page 180	Provides access to the operating system from the application.
“TextDocument object” on page 182	Encapsulates the text in a text layer.
“TextLayer object” on page 188	Represents a text layer within a composition.
“Viewer object” on page 189	Represents a Composition, Layer, or Footage panel.

Global functions

These globally available functions that are specific to After Effects. Any JavaScript object or function can call these functions, which allow you to display text in a small (3-line) area of the Info panel, and to convert numeric time values to and from string values.

Global function	Description
`clearOutput()`	Clears text from the Info panel.
`currentFormatToTime()`	Converts string time value to a numeric time value.
`timeToCurrentFormat()`	Converts a numeric time value to a string time value.
`write()`	Writes text to the Info panel, with no line break added.
`writeLn()`	Writes text to the Info panel, adding a line break at the end.
`isValid()`	When true, the specified object exists.

Additional global functions for standard user I/O (`alert`, `confirm`, and `prompt`) and static functions for file I/O, are defined by ExtendScript; for detailed reference information, see the *JavaScript Tools Guide* (available from the ExtendScript Toolkit's Help menu).

clearOutput() global function

`clearOutput()`

Description

Clears the output in the Info panel.

Parameters

None.

Returns

Nothing.

currentFormatToTime() global function

`currentFormatToTime(`*`formattedTime, fps, isDuration`*`)`

Description

Converts a formatted string for a frame time value to a number of seconds, given a specified frame rate. For example, if the formatted frame time value is 0:00:12 (the exact string format is determined by a project setting), and the frame rate is 24 fps, the time would be 0.5 seconds (12/24). If the frame rate is 30 fps, the time would be 0.4 seconds (12/30).

If the time is a duration, the frames are counted from 0. Otherwise, the frames are counted from the project's starting frame (see "Project displayStartFrame attribute" on page 117).

Parameters

`formattedTime`	The frame time value, a string specifying a number of frames in the project's current time display format.
`fps`	The frames-per-second, a floating-point value.

`isDuration`	Optional. When true, the time is a duration (measured from frame 0). When false (the default), the time is measured from the project's starting frame.

Returns

Floating-point value, the number of seconds.

isValid() global function

`isValid(`*`obj`*`)`

Description

Determines if the specified After Effects object (e.g., composition, layer, mask, etc.) still exists. Some operations, such as the `PropertyBase moveTo()` method, might invalidate existing variable assignments to related objects. This function allows you to test whether those assignments are still valid before attempting to access them.

Parameters

`obj`	The After Effects object to check for validity.

Returns

Boolean.

Example

```
var layer = app.project.activeItem.layer(1); // assume layer has three masks
alert(isValid(layer)); // displays "true"
var mask1 = layer.mask(1);
var mask2 = layer.mask(2);
var mask3 = layer.mask(3);
mask3.moveTo(1); // move the third mask to the top of the mask stack
alert(isValid(mask1)); // displays "false"; mask2 and mask3 do as well
```

timeToCurrentFormat() global function

`timeToCurrentFormat(`*`time, fps, isDuration`*`)`

Description

Converts a numeric time value (a number of seconds) to a frame time value; that is, a formatted string that shows which frame corresponds to that time, at the specified rate. For example, if the time is 0.5 seconds, and the frame rate is 24 fps, the frame would be 0:00:12 (when the project is set to display as timecode). If the frame rate is 30 fps, the frame would be 0:00:15. The format of the timecode string is determined by a project setting.

If the time is a duration, the frames are counted from 0. Otherwise, the frames are counted from the project's starting frame (see "Project displayStartFrame attribute" on page 117).

Parameters

`time`	The number of seconds, a floating-point value.
`fps`	The frames-per-second, a floating-point value.

isDuration	Optional. When true, the time is a duration (measured from frame 0). When false (the default), the time is measured from the project's starting frame.

Returns

String in the project's current time display format.

write() global function

`write(`*`text`*`)`

Description

Writes output to the Info panel, with no line break added.

Parameters

text	The string to display. Truncated if too long for the Info panel.

Returns

Nothing.

Example

```
write("This text appears in Info panel ");
write("with more on same line.");
```

writeLn() global function

`writeLn(`*`text`*`)`

Description

Writes output to the Info panel and adds a line break at the end.

Parameters

text	The string to display.

Returns

Nothing.

Example

```
writeln("This text appears on first line");
writeln("This text appears on second line");
```

Application object

`app`

Description

Provides access to objects and application settings within the After Effects application. The single global object is always available by its name, `app`.

Attributes of the Application object provide access to specific objects within After Effects. Methods of the Application object can create a project, open an existing project, control Watch Folder mode, purge memory, and quit the After Effects application. When the After Effects application quits, it closes the open project, prompting the user to save or discard changes as necessary, and creates a project file as necessary.

Attributes

Attribute	Reference	Description
`project`	"Application project attribute" on page 26 and "Project object" on page 114	The current After Effects project.
`isoLanguage`	"Application isoLanguage attribute" on page 22	The locale (language and region) in which the application is running.
`version`	"Application version attribute" on page 29	The version number of the After Effects application.
`buildName`	"Application buildName attribute" on page 19	The name of this build of the application.
`buildNumber`	"Application buildNumber attribute" on page 20	The number of this build of the application.
`isWatchFolder`	"Application isWatchFolder attribute" on page 23	When true, the local application is running in Watch Folder mode.
`isRenderEngine`	"Application isRenderEngine attribute" on page 23	When true, the local After Effects application is running as a render engine.
`settings`	"Application settings attribute" on page 28 and "RQItemCollection object" on page 169	Application settings that can be set via scripting.
`onError`	"Application onError attribute" on page 24	A callback function that is called when an error occurs in the application.
`exitCode`	"Application exitCode attribute" on page 22	A numeric status code used when executing a script externally (that is, from a command line or AppleScript). 0 if no error occurred. A positive number indicates an error that occurred while running the script.
`exitAfterLaunchAndEval`	"Application exitAfterLaunchAndEval attribute" on page 22	When true, the application remains open after running a script from the command line on Windows.
`saveProjectOnCrash`	"Application saveProjectOnCrash attribute" on page 27	When true, the project is saved if the application closes unexpectedly.
`memoryInUse`	"Application memoryInUse attribute" on page 23	Memory in use by this application.
`effects`	"Application effects attribute" on page 20	The effects available in the application.
`activeViewer`	"Application activeViewer attribute" on page 19	The currently focused or last-focused viewer panel.

Methods

Method	Reference	Description
`newProject()`	"Application newProject() method" on page 24	Creates a new project in After Effects.
`open()`	"Application open() method" on page 24	Opens a project or an Open Project dialog box.
`quit()`	"Application quit() method" on page 27	Quits the application.
`watchFolder()`	"Application watchFolder() method" on page 29	Starts Watch Folder mode; does not return until Watch Folder mode is turned off.
`pauseWatchFolder()`	"Application pauseWatchFolder() method" on page 26	Pauses a current watch-folder process.
`endWatchFolder()`	"Application endWatchFolder() method" on page 21	Ends a current watch-folder process.
`purge()`	"Application purge() method" on page 26	Purges a targeted type of cached information (replicates Purge options in the Edit menu).
`beginUndoGroup()`	"Application beginUndoGroup() method" on page 19	Groups the actions that follow it into a single undoable step.
`endUndoGroup()`	"Application endUndoGroup() method" on page 21	Ends an undo group; needed only when a script contains more than one undo group.
`beginSuppressDialogs()`	"Application beginSuppressDialogs() method" on page 19	Begins suppression of dialogs in the user interface.
`endSuppressDialogs()`	"Application endSuppressDialogs() method" on page 21	Ends suppression of dialogs in the user interface.
`setMemoryUsageLimits()`	"Application setMemoryUsageLimits() method" on page 28	Sets memory usage limits as in the Memory & Cache preferences area.
`setSavePreferencesOnQuit()`	"Application setSavePreferencesOnQuit() method" on page 28	Sets whether preferences are saved when the application is quit.
`activate()`	"Application activate() method" on page 18	Brings the After Effects main window to the front of the screen.
`scheduleTask()`	"Application scheduleTask() method" on page 27	Schedules a JavaScript script for delayed execution.
`cancelTask()`	"Application cancelTask() method" on page 20	Cancels a scheduled task.
`parseSwatchFile()`	"Application parseSwatchFile() method" on page 25	Loads a color swatch from an Adobe Swatch Exchange (ASE) file.

Application activate() method

```
app.activate()
```

Description

Opens the application main window if it is minimized or iconified, and brings it to the front of the desktop.

Parameters

None.

Returns

Nothing.

Application activeViewer attribute

`app.activeViewer`

Description

The Viewer object for the currently focused or active-focused viewer (Composition, Layer, or Footage) panel. Returns null if no viewers are open.

Type

Viewer object; read-only.

Application beginSuppressDialogs() method

`app.beginSuppressDialogs()`

Description

Begins suppression of script error dialog boxes in the user interface. Use `endSuppressDialogs()` to resume the display of error dialogs. See “Application endSuppressDialogs() method” on page 21.

Parameters

None.

Returns

Nothing.

Application beginUndoGroup() method

`app.beginUndoGroup(`*`undoString`*`)`

Description

Marks the beginning of an undo group, which allows a script to logically group all of its actions as a single undoable action (for use with the Edit > Undo/Redo menu items). Use the `endUndoGroup()` method to mark the end of the group. (See “Application endUndoGroup() method” on page 21.)

`beginUndoGroup()` and `endUndoGroup()` pairs can be nested. Groups within groups become part of the larger group, and will undo correctly. In this case, the names of inner groups are ignored.

Parameters

`undoString`	The text that will appear for the Undo command in the Edit menu (that is, “Undo `<undoString>`”)

Returns

Nothing.

Application buildName attribute

`app.buildName`

Description

The name of the build of After Effects being run, used internally by Adobe for testing and troubleshooting.

Type

String; read-only.

Application buildNumber attribute

`app.buildNumber`

Description

The number of the build of After Effects being run, used internally by Adobe for testing and troubleshooting.

Type

Integer; read-only.

Application cancelTask() method

`app.cancelTask(`*`taskID`*`)`

Description

Removes the specified task from the queue of tasks scheduled for delayed execution.

Parameters

`taskID`	An integer that identifies the task, as returned by `app.scheduleTask()`.

Returns

Nothing.

Application effects attribute

`app.effects`

Description

The effects available in the application.

Type

Array, with each element containing the following properties; read-only:

`displayName`	String representing the localized display name of the effect as seen in the Effect menu.
`category`	String representing the localized category label as seen in the Effect menu. This can be "" for synthetic effects that aren't normally shown to the user.
`matchName`	String representing the internal unique name for the effect. This name does not change between versions of After Effects. Use this value to apply the effect.

Example

```
var effectName = app.effects[12].displayName;
```

Application endSuppressDialogs() method

`app.endSuppressDialogs(`*`alert`*`)`

Description

Ends the suppression of script error dialog boxes in the user interface. Error dialogs are displayed by default; call this method only if `beginSuppressDialogs()` has previously been called. See "Application beginSuppress-Dialogs() method" on page 19.

Parameters

`alert`	Boolean; when true, errors that have occurred following the call to `beginSuppressDialogs()` are displayed in a dialog box.

Returns

Nothing.

Application endUndoGroup() method

`app.endUndoGroup()`

Description

Marks the end of an undo group begun with the `app.beginUndoGroup()` method. You can use this method to place an end to an undo group in the middle of a script, should you wish to use more than one undo group for a single script.

If you are using only a single undo group for a given script, you do not need to use this method; in its absence at the end of a script, the system will close the undo group automatically.

Calling this method without having set a `beginUndoGroup()` method yields an error.

Parameters

None.

Returns

Nothing.

Application endWatchFolder() method

`app.endWatchFolder()`

Description

Ends Watch Folder mode.

Parameters

None.

Returns

Nothing.

See also

"Application watchFolder() method" on page 29
"Application parseSwatchFile() method" on page 25
"Application isWatchFolder attribute" on page 23

Application exitAfterLaunchAndEval attribute

`app.exitAfterLaunchAndEval`

Description

This attribute is used only when executing a script from a command line on Windows. When the application is launched from the command line, the `-r` or `-s` command line flag causes the application to run a script (from a file or from a string, respectively).

If this attribute is set to true, After Effects will exit after the script is run; if it is false, the application will remain open.

This attribute only has an effect when After Effects is run from the Windows command line. It has no effect in Mac OS.

Type

Boolean; read/write.

Application exitCode attribute

`app.exitCode`

Description

A numeric status code used when executing a script externally (that is, from a command line or AppleScript).

- In Windows, the value is returned on the command line when After Effects was launched on the command line (using the `afterfx` or `afterfx -m` command), and a script was specified with the `-r` or `-s` option.
- in Mac OS, the value is returned as the AppleScript `DoScript` result for each script.

In both Mac OS and Windows, the value is set to 0 (`EXIT_SUCCESS`) at the beginning of each script evaluation. In the event of an error while the script is running, the script can set this to a positive integer that indicates what error occurred.

Type

Integer; read/write.

Example

```
app.exitCode = 2; //on quit, if value is 2, an error has occurred
```

Application isoLanguage attribute

`app.isoLanguage`

Description

A string indicating the locale (language and regional designations) After Effects is running.

NOTE: `$.locale` *returns the operating system language, not the language of the After Effects application.*

Type

String; read-only. Some common values include:

- `en_US` for English (United States)
- `de_DE` for German (Germany)
- `es_ES` for Spanish (Spain)
- `fr_FR` for French (France)
- `it_IT` for Italian (Italy)
- `ja_JP` for Japanese (Japan)
- `ko_KR` for Korean (Korea)

Example

```
var lang = app.isoLanguage;
if (lang == "en_US")
  alert("After Effects is running in English.");
else if (lang == "fr_FR")
  alert("After Effects is running in French.");
else
  alert("After Effects is not running in English or French.");
```

Application isRenderEngine attribute

`app.isRenderEngine`

Description

True if After Effects is running as a render engine.

Type

Boolean; read-only.

Application isWatchFolder attribute

`app.isWatchFolder`

Description

True if the Watch Folder dialog box is currently displayed and the application is currently watching a folder for rendering.

Type

Boolean; read-only.

Application memoryInUse attribute

`app.memoryInUse`

Description

The number of bytes of memory currently used by this application.

Type

Number; read-only.

Application newProject() method

`app.newProject()`

Description

Creates a new project in After Effects, replicating the File > New > New Project menu command.

If the current project has been edited, the user is prompted to save it. If the user cancels out of the Save dialog box, the new project is not created and the method returns null. Use `app.project.close(CloseOptions.DO_NOT_SAVE_CHANGES)` to close the current project before opening a new one. See “Project close() method” on page 116.

Parameters

None.

Returns

A new Project object, or null if no new project is created.

Example

```
app.project.close(CloseOptions.DO_NOT_SAVE_CHANGES);
app.newProject();
```

Application onError attribute

`app.onError`

Description

The name of a callback function that is called when an error occurs. By creating a function and assigning it to this attribute, you can respond to errors systematically; for example, you can close and restart the application, noting the error in a log file if it occurred during rendering. See “RenderQueue render() method” on page 161.

The callback function is passed the error string and a severity string. It should not return any value.

Type

A function name string, or null if no function is assigned; read/write.

Example

```
function err(errString) {
  alert(errString);
  }
app.onError = err;
```

Application open() method

```
app.open()
app.open(file)
```

Description

Opens a project.

Parameters

file	Optional. An ExtendScript File object for the project file to open. If not supplied, the method prompts the user to select a project file.

Returns

A new Project object for the specified project, or null if the user cancels the Open dialog box.

Example

```
var my_file = new File("../my_folder/my_test.aep");
if (my_file.exists){
  new_project = app.open(my_file);
  if (new_project){
      alert(new_project.file.name);
    }
  }
```

Application parseSwatchFile() method

app.parseSwatchFile(*file*)

Description

Loads color swatch data from an Adobe Swatch Exchange (ASE) file.

Parameters

file	The file specification, an ExtendScript File object.

Returns

The swatch data, in this format:

data.majorVersion data.minorVersion	The ASE version number.
data.values	An array of SwatchValue.
SwatchValue.type	One of "RGB", "CMYK", "LAB", "Gray"
SwatchValue.r SwatchValue.g SwatchValue.b	When type = "RGB", the color values in the range [0.0..1.0]. 0, 0, 0 is Black.
SwatchValue.c SwatchValue.m SwatchValue.y SwatchValue.k	When type = "CMYK", the color values in the range [0.0..1.0]. 0, 0, 0, 0 is White.

`SwatchValue.L`
`SwatchValue.a`
`SwatchValue.b`
	When `type = "LAB"`, the color values.
	`L` is in the range [0.0..1.0]. `a` and `b` are in the range [-128.0..+128.0]
	0, 0, 0 is Black.

`SwatchValue.value`	When `type = "Gray"`, the `value` range is [0.0..1.0]. 0.0 is Black.

Application pauseWatchFolder() method

`app.pauseWatchFolder(`*`pause`*`)`

Description

Pauses or resumes the search of the target watch folder for items to render.

Parameters

`pause`	True to pause, false to resume.

Returns

Nothing.

See also

"Application isWatchFolder attribute" on page 23
"Application watchFolder() method" on page 29
"Application endWatchFolder() method" on page 21

Application project attribute

`app.project`

Description

The project that is currently loaded. See "Project object" on page 114.

Type

Project object; read-only.

Application purge() method

`app.purge(`*`target`*`)`

Description

Purges unused data of the specified types from memory. Replicates the Purge options in the Edit menu.

Parameters

target	The type of elements to purge from memory; a PurgeTarget enumerated value, one of: • PurgeTarget.ALL_CACHES: Purges all data that After Effects has cached to physical memory. • PurgeTarget.UNDO_CACHES: Purges all data saved in the undo cache. • PurgeTarget.SNAPSHOT_CACHES: Purges all data cached as composition/layer snapshots. • PurgeTarget.IMAGE_CACHES: Purges all saved image data.

Returns

Nothing.

Application quit() method

`app.quit()`

Description

Quits the After Effects application.

Parameters

None.

Returns

Nothing.

Application saveProjectOnCrash attribute

`app.saveProjectOnCrash`

Description

When true (the default), After Effects attempts to display a dialog box that allows you to save the current project if an error causes the application to quit unexpectedly. Set to false to suppress this dialog box and quit without saving.

Type

Boolean; read/write.

Application scheduleTask() method

app.scheduleTask(*stringToExecute, delay, repeat*)

Description

Schedules the specified JavaScript for delayed execution.

Parameters

stringToExecute	A string containing JavaScript to be executed.
delay	A number of milliseconds to wait before executing the JavaScript. A floating-point value.

`repeat`	When true, execute the script repeatedly, with the specified delay between each execution. When false the script is executed only once.

Returns

Integer, a unique identifier for this task, which can be used to cancel it with `app.cancelTask()`.

Application setMemoryUsageLimits() method

`app.setMemoryUsageLimits(`*`imageCachePercentage, maximumMemoryPercentage`*`)`

Description

Sets memory usage limits as in the Memory & Cache preferences area. For both values, if installed RAM is less than a given amount (*n* gigabytes), the value is a percentage of the installed RAM, and is otherwise a percentage of *n*. The value of *n* is: 2 GB for 32-bit Windows, 4 GB for 64-bit Windows, 3.5 GB for Mac OS.

Parameters

`imageCachePercentage`	Floating-point value, the percentage of memory assigned to image cache.
`maximumMemoryPercentage`	Floating-point value, the maximum usable percentage of memory.

Returns

Nothing.

Application setSavePreferencesOnQuit() method

`app.setSavePreferencesOnQuit(`*`doSave`*`)`

Description

Set or clears the flag that determines whether preferences are saved when the application is closed.

Parameters

`doSave`	When true, preferences saved on quit, when false they are not.

Returns

Nothing.

Application settings attribute

`app.settings`

Description

The currently loaded settings. See “Settings object” on page 170.

Type

Settings object; read-only.

Application version attribute

`app.version`

Description

An alphanumeric string indicating which version of After Effects is running.

Type

String; read-only.

Example

```
var ver = app.version;
alert("This machine is running version " + ver + " of After Effects.");
```

Application watchFolder() method

app.watchFolder(*folder_object_to_watch*)

Description

Starts a Watch Folder (network rendering) process pointed at a specified folder.

Parameters

`folder_object_to_watch`	The ExtendScript Folder object for the folder to watch.

Returns

Nothing.

Example

```
var theFolder = new Folder("c:/tool");
app.watchFolder(theFolder);
```

See also

"Application endWatchFolder() method" on page 21
"Application parseSwatchFile() method" on page 25
"Application isWatchFolder attribute" on page 23

AVItem object

app.project.item(*index*)

Description

The AVItem object provides access to attributes and methods of audio/visual files imported into After Effects.

- AVItem is a subclass of Item. All methods and attributes of Item, in addition to those listed below, are available when working with AVItem. See "Item object" on page 78.
- AVItem is the base class for both CompItem and FootageItem, so AVItem attributes and methods are also available when working with CompItem and FootageItem objects. See "CompItem object" on page 52 and "FootageItem object" on page 65.

Attributes

Attribute	Reference	Description
name	"AVItem name attribute" on page 33	The name of the object as shown in the Project panel.
width	"AVItem width attribute" on page 37	The width of the item.
height	"AVItem height attribute" on page 32	The height of the item.
pixelAspect	"AVItem pixelAspect attribute" on page 33	The pixel aspect ratio of the item.
frameRate	"AVItem frameRate attribute" on page 32	The frame rate of the item.
frameDuration	"AVItem frameDuration attribute" on page 31	The frame duration for the item.
duration	"AVItem duration attribute" on page 31	The total duration of the item.
useProxy	"AVItem useProxy attribute" on page 36	When true, a proxy source is used for this item.
proxySource	"AVItem proxySource attribute" on page 34	The FootageItem object used as proxy for the item.
time	"AVItem time attribute" on page 36	Current time of the item.
usedIn	"AVItem usedIn attribute" on page 36	The CompItem objects that use this item.
hasVideo	"AVItem hasVideo attribute" on page 32	When true, the item has a video component.
hasAudio	"AVItem hasAudio attribute" on page 32	When true, the item has an audio component.
footageMissing	"AVItem footageMissing attribute" on page 31	When true, the item cannot be found or is a placeholder.

Methods

Method	Reference	Description
setProxy()	"AVItem setProxy() method" on page 34	Sets a proxy for the item.
setProxyWithSequence()	"AVItem setProxyWithSequence() method" on page 35	Sets a sequence as a proxy for the item.
setProxyWithSolid()	"AVItem setProxyWithSolid() method" on page 35	Sets a solid as a proxy for the item.
setProxyWithPlaceholder()	"AVItem setProxyWithPlaceholder() method" on page 35	Sets a placeholder as a proxy for the item.
setProxyToNone()	"AVItem setProxyToNone() method" on page 34	Removes the proxy for the item.

AVItem duration attribute

`app.project.item(index).duration`

Description

Returns the duration, in seconds, of the item. Still footage items have a duration of 0.

- In a CompItem, the value is linked to the `duration` of the composition, and is read/write.
- In a FootageItem, the value is linked to the `duration` of the `mainSource` object, and is read-only.

Type

Floating-point value in the range [0.0..10800.0]; read/write for a CompItem; otherwise, read-only.

AVItem footageMissing attribute

`app.project.item(index).footageMissing`

Description

When true, the AVItem is a placeholder, or represents footage with a source file that cannot be found. In this case, the path of the missing source file is in the `missingFootagePath` attribute of the footage item's source-file object. See "FootageItem mainSource attribute" on page 66 and "FileSource missingFootagePath attribute" on page 61.

Type

Boolean; read-only.

AVItem frameDuration attribute

`app.project.item(index).frameDuration`

Description

Returns the length of a frame for this AVItem, in seconds. This is the reciprocal of `frameRate`. When set, the reciprocal is automatically set as a new `frameRate` value.

This attribute returns the reciprocal of the `frameRate`, which may not be identical to a value you set, if that value is not evenly divisible into 1.0 (for example, 0.3). Due to numerical limitations, (1 / (1 / 0.3)) is close to, but not exactly, 0.3.

If the AVItem is a FootageItem, this value is linked to the `mainSource`, and is read-only. To change it, set the `conformFrameRate` of the `mainSource` object. This sets both the `frameRate` and `frameDuration` of the FootageItem.

Type

Floating-point value in the range [1/99.. 1.0]; read-only for a FootageItem, otherwise read/write.

AVItem frameRate attribute

`app.project.item(index).frameRate`

Description

The frame rate of the AVItem, in frames-per-second. This is the reciprocal of the `frameDuration`. When set, the reciprocal is automatically set as a new `frameDuration` value.

- In a CompItem, the value is linked to the `frameRate` of the composition, and is read/write.
- In a FootageItem, the value is linked to the `frameRate` of the `mainSource` object, and is read-only. To change it, set the `conformFrameRate` of the `mainSource` object. This sets both the `frameRate` and `frameDuration` of the FootageItem.

Type

Floating-point value in the range [1.0..99.0]; read-only for a FootageItem, otherwise read/write.

AVItem hasAudio attribute

`app.project.item(index).hasAudio`

Description

When true, the AVItem has an audio component.

- In a CompItem, the value is linked to the composition.
- In a FootageItem, the value is linked to the `mainSource` object.

Type

Boolean; read-only.

AVItem hasVideo attribute

`app.project.item(index).hasVideo`

Description

When true, the AVItem has an video component.

- In a CompItem, the value is linked to the composition.
- In a FootageItem, the value is linked to the `mainSource` object.

Type

Boolean; read-only.

AVItem height attribute

`app.project.item(index).height`

Description

The height of the item in pixels.

- In a CompItem, the value is linked to the composition, and is read/write.

- In a FootageItem, the value is linked to the `mainSource` object, and is read/write only if the `mainSource` object is a SolidSource. Otherwise, it is read-only.

Type

Integer in the range [1...30000]; read/write, except as noted.

AVItem name attribute

`app.project.item(index).name`

Description

The name of the item, as shown in the Project panel.

- In a FootageItem, the value is linked to the `mainSource` object. If the `mainSource` object is a FileSource, this value controls the display name in the Project panel, but does not affect the file name.

Type

String; read/write.

AVItem pixelAspect attribute

`app.project.item(index).pixelAspect`

Description

The pixel aspect ratio (PAR) of the item.

- In a CompItem, the value is linked to the composition.
- In a FootageItem, the value is linked to the `mainSource` object.

The value you retrieve after setting may be slightly different from the value you supplied. The following table compares the value as it appears in the UI with the more-accurate value returned by this attribute.

PAR in the After Effects UI	PAR returned by the pixelAspect attribute
0.91	0.90909090909091
1	1
1.5	1.5
1.09	1.09401709401709
1.21	1.21212121212121
1.33	1.33333333333333
1.46	1.45868945868946
2	2

Type

Floating-point value, in the range [0.01..100.0]; read/write.

AVItem proxySource attribute

`app.project.item(index).proxySource`

Description

The FootageSource being used as a proxy. The attribute is read-only; to change it, call any of the AVItem methods that change the proxy source: `setProxy()`, `setProxyWithSequence()`, `setProxyWithSolid()`, or `setProxyWithPlaceholder()`.

Type

FootageSource object; read-only.

AVItem setProxy() method

`app.project.item(index).setProxy(`*`file`*`)`

Description

Sets a file as the proxy of this AVItem. Loads the specified file into a new FileSource object, sets this as the value of the `proxySource` attribute, and sets `useProxy` to true. It does not preserve the interpretation parameters, instead using the user preferences. If the file has an unlabeled alpha channel, and the user preference says to ask the user what to do, the method estimates the alpha interpretation, rather than asking the user.

This differs from setting a FootageItem's main source, but both actions are performed as in the user interface.

Parameters

`file`	An ExtendScript File object for the file to be used as a proxy.

Returns

None.

AVItem setProxyToNone() method

`app.project.item(index).setProxyToNone()`

Description

Removes the proxy from this AVItem, sets the value of `proxySource` to null, and sets the value of `useProxy` to `false`.

Parameters

None.

Returns

Nothing.

AVItem setProxyWithPlaceholder() method

`app.project.item(index).setProxyWithPlaceholder(`*`name, width, height, frameRate, duration`*`)`

Description

Creates a PlaceholderSource object with specified values, sets this as the value of the `proxySource` attribute, and sets `useProxy` to true. It does not preserve the interpretation parameters, instead using the user preferences.

NOTE: There is no direct way to set a placeholder as a proxy in the user interface; this behavior occurs when a proxy has been set and then moved or deleted.

Parameters

`name`	A string containing the name of the new object.
`width, height`	The pixel dimensions of the placeholder, an integer in the range [4..30000].
`frameRate`	The frames-per-second, an integer in the range [1..99].
`duration`	The total length in seconds, up to 3 hours. An integer in the range [0.0..10800.0].

Returns

Nothing.

AVItem setProxyWithSequence() method

`app.project.item(index).setProxyWithSequence(`*`file, forceAlphabetical`*`)`

Description

Sets a sequence of files as the proxy of this AVItem, with the option of forcing alphabetical order. Loads the specified file sequence into a new FileSource object, sets this as the value of the `proxySource` attribute, and sets `useProxy` to true. It does not preserve the interpretation parameters, instead using the user preferences. If any file has an unlabeled alpha channel, and the user preference says to ask the user what to do, the method estimates the alpha interpretation, rather than asking the user.

Parameters

`file`	An ExtendScript File object for the first file in the sequence.
`forceAlphabetical`	When true, use the “Force alphabetical order” option.

Returns

Nothing.

AVItem setProxyWithSolid() method

`app.project.item(index).setProxyWithSolid(`*`color, name, width, height, pixelAspect`*`)`

Description

Creates a SolidSource object with specified values, sets this as the value of the `proxySource` attribute, and sets `useProxy` to true. It does not preserve the interpretation parameters, instead using the user preferences.

NOTE: There is no way, using the user interface, to set a solid as a proxy; this feature is available only through scripting.

Parameters

`color`	The color of the solid, an array of 3 floating-point values, [R, G, B], in the range [0.0..1.0].
`name`	A string containing the name of the new object.
`width, height`	The pixel dimensions of the placeholder, an integer in the range [1...30000].
`pixelAspect`	The pixel aspect of the solid, a floating-point value in the range [0.01... 100.0].

Returns

Nothing.

AVItem time attribute

`app.project.item(index).time`

Description

The current time of the item when it is being previewed directly from the Project panel. This value is a number of seconds. Use the global method `timeToCurrentFormat` to convert it to a string value that expresses the time in terms of frames; see "timeToCurrentFormat() global function" on page 15.

It is an error to set this value for a FootageItem whose `mainSource` is still (*`item`*`.mainSource.isStill` is true).

Type

Floating-point value; read/write.

AVItem usedIn attribute

`app.project.item(index).usedIn`

Description

All the compositions that use this AVItem.

Note that upon retrieval, the array value is copied, so it is not automatically updated. If you get this value, then add this item into another composition, you must retrieve the value again to get an array that includes the new item.

Type

Array of CompItem objects; read-only.

AVItem useProxy attribute

`app.project.item(index).useProxy`

Description

When true, a proxy is used for the item. It is set to true by all the `SetProxy` methods, and to false by the `SetProxyToNone()` method.

Type

Boolean; read/write.

AVItem width attribute

`app.project.item(index).width`

Description

The width of the item, in pixels.

- In a CompItem, the value is linked to the composition, and is read/write.
- In a FootageItem, the value is linked to the `mainSource` object, and is read/write only if the `mainSource` object is a SolidSource. Otherwise, it is read-only.

Type

Integer in the range [1...30000]; read/write, except as noted.

AVLayer object

`app.project.item(index).layer(`*`index`*`)`

Description

The AVLayer object provides an interface to those layers that contain AVItem objects (composition layers, footage layers, solid layers, text layers, and sound layers).

- AVLayer is a subclass of Layer. All methods and attributes of Layer, in addition to those listed below, are available when working with AVLayer. See “Layer object” on page 86.
- AVLayer is a base class for TextLayer, so AVLayer attributes and methods are available when working with TextLayer objects. See “TextLayer object” on page 188.

AE Properties

Different types of layers have different AE properties. AVLayer has the following properties and property groups:

```
Marker
Time Remap
Motion Trackers
Masks
Effects
Transform
    Anchor Point
    Position
    Scale
    Orientation
    X Rotation
    Y Rotation
    Rotation
    Opacity
Layer Styles
Geometry Options    // Ray-traced 3D
Material Options
    Casts Shadows
    Light Transmission
    Accepts Shadows
    Accepts Lights
    Appears in Reflections    // Ray-traced 3D
    Ambient
    Diffuse
    Specular Intensity
    Specular Shininess
    Metal
    Reflection Intensity    // Ray-traced 3D
    Reflection Sharpness    // Ray-traced 3D
    Reflection Rolloff    // Ray-traced 3D
    Transparency    // Ray-traced 3D
    Transparency Rolloff    // Ray-traced 3D
    Index of Refraction    // Ray-traced 3D
Audio
    Audio Levels
```

Example

If the first item in the project is a CompItem, and the first layer of that CompItem is an AVLayer, the following sets the layer `quality`, `startTime`, and `inPoint`.

```
var firstLayer = app.project.item(1).layer(1);
firstLayer.quality = LayerQuality.BEST;
firstLayer.startTime = 1;
firstLayer.inPoint = 2;
```

Attributes

Attribute	Reference	Description
`source`	"AVLayer source attribute" on page 47	The source item for this layer.
`isNameFromSource`	"AVLayer isNameFromSource attribute" on page 46	When true, the layer has no expressly set name, but contains a named source.
`height`	"AVLayer height attribute" on page 45	The height of the layer.
`width`	"AVLayer width attribute" on page 49	The width of the layer.
`audioEnabled`	"AVLayer audioEnabled attribute" on page 41	When true, the layer's audio is enabled.
`motionBlur`	"AVLayer motionBlur attribute" on page 46	When true, the layer's motion blur is enabled.
`effectsActive`	"AVLayer effectsActive attribute" on page 44	When true, the layer's effects are active.
`adjustmentLayer`	"AVLayer adjustmentLayer attribute" on page 40	When true, this is an adjustment layer.
`guideLayer`	"AVLayer guideLayer attribute" on page 45	When true, this is a guide layer.
`threeDLayer`	"AVLayer threeDLayer attribute" on page 48	When true, this is a 3D layer.
`threeDPerChar`	"AVLayer threeDPerChar attribute" on page 48	When true, 3D is set on a per-character basis in this text layer.
`environmentLayer`	"AVLayer environmentLayer attribute" on page 44	When true, this is an environment layer.
`canSetCollapseTransformation`	"AVLayer canSetCollapseTransformation attribute" on page 43	When true, it is legal to change the value of `collapseTransformation`.
`collapseTransformation`	"AVLayer collapseTransformation attribute" on page 44	When true, collapse transformation is on.
`frameBlending`	"AVLayer frameBlending attribute" on page 44	When true, frame blending is enabled.
`frameBlendingType`	"AVLayer frameBlendingType attribute" on page 44	The type of frame blending for the layer.
`canSetTimeRemapEnabled`	"AVLayer canSetTimeRemapEnabled attribute" on page 43	When true, it is legal to change the value of `timeRemapEnabled`.
`timeRemapEnabled`	"AVLayer timeRemapEnabled attribute" on page 48	When true, time remapping is enabled on this layer.
`hasAudio`	"AVLayer hasAudio attribute" on page 45	When true, the layer contains an audio component.

Attribute	Reference	Description
audioActive	“AVLayer audioActive attribute” on page 40	When true, the layer's audio is active at the current time.
blendingMode	“AVLayer blendingMode attribute” on page 42	The blending mode of the layer.
preserveTransparency	“AVLayer preserveTransparency attribute” on page 47	When true, preserve transparency is enabled.
trackMatteType	“AVLayer trackMatteType attribute” on page 49	if layer has a track matte, specifies the way it is applied.
isTrackMatte	“AVLayer isTrackMatte attribute” on page 46	When true, this layer is being used as a track matte for the layer below it.
hasTrackMatte	“AVLayer hasTrackMatte attribute” on page 45	When true, the layer above is being used as a track matte on this layer.
quality	“AVLayer quality attribute” on page 47	The layer quality setting.
autoOrient	“AVLayer autoOrient attribute” on page 41	The type of automatic orientation for the layer.

Methods

Method	Reference	Description
audioActiveAtTime()	“AVLayer audioActiveAtTime() method” on page 41	Reports whether this layer's audio is active at a given time.
calculateTransformFromPoints()	“AVLayer calculateTransform-FromPoints() method” on page 43	Calculates a transformation from a set of points in this layer.
replaceSource()	“AVLayer replaceSource() method” on page 47	Changes the source item for this layer.
sourceRectAtTime()	“AVLayer sourceRectAtTime() method” on page 48	Retrieves the source rectangle of a layer.
openInViewer()	“AVLayer openInViewer() method” on page 46	Opens the layer in a Layer panel.

AVLayer adjustmentLayer attribute

app.project.item(index).layer(*index*).adjustmentLayer

Description

True if the layer is an adjustment layer.

Type

Boolean; read/write.

AVLayer audioActive attribute

app.project.item(index).layer(*index*).audioActive

Description

True if the layer's audio is active at the current time.

For this value to be true, `audioEnabled` must be true, no other layer with audio may be soloing unless this layer is soloed too, and the time must be between the `inPoint` and `outPoint` of this layer.

Type

Boolean; read-only.

AVLayer audioActiveAtTime() method

app.project.item(index).layer(*index*).audioActiveAtTime(*time*)

Description

Returns true if this layer's audio will be active at the specified time.

For this method to return true, `audioEnabled` must be true, no other layer with audio may be soloing unless this layer is soloed too, and the time must be between the `inPoint` and `outPoint` of this layer.

Parameters

time	The time, in seconds. A floating-point value.

Returns

Boolean.

AVLayer audioEnabled attribute

app.project.item(index).layer(*index*).audioEnabled

Description

When true, the layer's audio is enabled. This value corresponds to the audio toggle switch in the Timeline panel.

Type

Boolean; read/write.

AVLayer autoOrient attribute

app.project.item(index).layer(*index*).autoOrient

Description

The type of automatic orientation to perform for the layer.

Type

An `AutoOrientType` enumerated value; read/write. One of:

AutoOrientType.ALONG_PATH	Layer faces in the direction of the motion path.
AutoOrientType.CAMERA_OR_POINT_OF_INTEREST	Layer always faces the active camera or points at its point of interest.
AutoOrientType.CHARACTERS_TOWARD_CAMERA	Each character in a per-character 3D text layer automatically faces the active camera.

AutoOrientType.NO_AUTO_ORIENT	Layer rotates freely, independent of any motion path, point of interest, or other layers.

AVLayer blendingMode attribute

app.project.item(index).layer(*index*).blendingMode

Description

The blending mode of the layer.

Type

A BlendingMode enumerated value; read/write. One of:

BlendingMode.ADD
BlendingMode.ALPHA_ADD
BlendingMode.CLASSIC_COLOR_BURN
BlendingMode.CLASSIC_COLOR_DODGE
BlendingMode.CLASSIC_DIFFERENCE
BlendingMode.COLOR
BlendingMode.COLOR_BURN
BlendingMode.COLOR_DODGE
BlendingMode.DANCING_DISSOLVE
BlendingMode.DARKEN
BlendingMode.DARKER_COLOR
BlendingMode.DIFFERENCE
BlendingMode.DISSOLVE
BlendingMode.EXCLUSION
BlendingMode.HARD_LIGHT
BlendingMode.HARD_MIX
BlendingMode.HUE
BlendingMode.LIGHTEN
BlendingMode.LIGHTER_COLOR
BlendingMode.LINEAR_BURN
BlendingMode.LINEAR_DODGE
BlendingMode.LINEAR_LIGHT
BlendingMode.LUMINESCENT_PREMUL
BlendingMode.LUMINOSITY
BlendingMode.MULTIPLY
BlendingMode.NORMAL
BlendingMode.OVERLAY
BlendingMode.PIN_LIGHT
BlendingMode.SATURATION
BlendingMode.SCREEN
BlendingMode.SILHOUETE_ALPHA
BlendingMode.SILHOUETTE_LUMA
BlendingMode.SOFT_LIGHT
BlendingMode.STENCIL_ALPHA
BlendingMode.STENCIL_LUMA
BlendingMode.VIVID_LIGHT

AVLayer calculateTransformFromPoints() method

app.project.item(index).layer(*index*).calculateTransformFromPoints(*pointTopLeft, pointTopRight, point-BottomRight*)

Description

Calculates a transformation from a set of points in this layer.

Parameters

pointTopLeft	The top left point coordinates in the form of an array, [x, y, z].
pointTopRight	The top right point coordinates in the form of an array, [x, y, z].
pointBottomRight	The bottom right point coordinates in the form of an array, [x, y, z].

Returns

An Object with the transformation properties set.

Example

```
var newLayer = comp.layers.add(newFootage);
newLayer.threeDLayer = true;

newLayer.blendingMode = BlendingMode.ALPHA_ADD;
var transform = newLayer.calculateTransformFromPoints(tl, tr, bl);
for(var sel in transform) {
  newLayer.transform[sel].setValue(transform[sel]);
  }
```

AVLayer canSetCollapseTransformation attribute

app.project.item(index).layer(*index*).canSetCollapseTransformation

Description

True if it is legal to change the value of the `collapseTransformation` attribute on this layer.

Type

Boolean; read-only.

AVLayer canSetTimeRemapEnabled attribute

app.project.item(index).layer(*index*).canSetTimeRemapEnabled

Description

True if it is legal to change the value of the `timeRemapEnabled` attribute on this layer.

Type

Boolean; read-only.

AVLayer collapseTransformation attribute

`app.project.item(index).layer(index).collapseTransformation`

Description

True if collapse transformation is on for this layer.

Type

Boolean; read/write.

AVLayer effectsActive attribute

`app.project.item(index).layer(index).effectsActive`

Description

True if the layer's effects are active, as indicated by the <f> icon next to it in the user interface.

Type

Boolean; read/write.

AVLayer environmentLayer attribute

`app.project.item(index).layer(index).environmentLayer`

Description

True if this is an environment layer in a Ray-traced 3D composition. Setting this attribute to true automatically makes the layer 3D (`threeDLayer` becomes true).

Type

Boolean; read/write.

AVLayer frameBlending attribute

`app.project.item(index).layer(index).frameBlending`

Description

True if frame blending is enabled for the layer.

Type

Boolean; read-only.

AVLayer frameBlendingType attribute

`app.project.item(index).layer(index).frameBlendingType`

Description

The type of frame blending to perform when frame blending is enabled for the layer.

Type

A `FrameBlendingType` enumerated value; read/write. One of:

```
FrameBlendingType.FRAME_MIX
FrameBlendingType.NO_FRAME_BLEND
FrameBlendingType.PIXEL_MOTION
```

AVLayer guideLayer attribute

`app.project.item(index).layer(`*`index`*`).guideLayer`

Description

True if the layer is a guide layer.

Type

Boolean; read/write.

AVLayer hasAudio attribute

`app.project.item(index).layer(`*`index`*`).hasAudio`

Description

True if the layer contains an audio component, regardless of whether it is audio-enabled or soloed.

Type

Boolean; read-only.

AVLayer hasTrackMatte attribute

`app.project.item(index).layer(`*`index`*`).hasTrackMatte`

Description

True if the layer in front of this layer is being used as a track matte on this layer. When true, this layer's `trackMatteType` value controls how the matte is applied.

Type

Boolean; read-only.

AVLayer height attribute

`app.project.item(index).layer(`*`index`*`).height`

Description

The height of the layer in pixels.

Type

Floating-point; read-only.

AVLayer isNameFromSource attribute

`app.project.item(index).layer(index).isNameFromSource`

Description

True if the layer has no expressly set name, but contains a named source. In this case, `layer.name` has the same value as `layer.source.name`.

False if the layer has an expressly set name, or if the layer does not have a source.

Type

Boolean; read-only.

AVLayer isTrackMatte attribute

`app.project.item(index)layer(index).isTrackMatte`

Description

True if this layer is being used as a track matte for the layer behind it.

Type

Boolean; read-only.

AVLayer motionBlur attribute

`app.project.item(index).layer(index).motionBlur`

Description

True if motion blur is enabled for the layer.

Type

Boolean; read/write.

AVLayer openInViewer() method

`app.project.item(index).layer(index).openInViewer()`

Description

Opens the layer in a Layer panel, and moves the Layer panel to front and gives it focus.

Parameters

None.

Returns

Viewer object for the Layer panel, or null if the layer could not be opened (e.g., for text or shape layers, which cannot be opened in the Layer panel).

AVLayer preserveTransparency attribute

`app.project.item(index).layer(index).preserveTransparency`

Description

True if preserve transparency is enabled for the layer.

Type

Boolean; read/write.

AVLayer quality attribute

`app.project.item(index).layer(index).quality`

Description

The quality with which this layer is displayed.

Type

A `LayerQuality` enumerated value; read/write. One of:

```
LayerQuality.BEST
LayerQuality.DRAFT
LayerQuality.WIREFRAME
```

AVLayer replaceSource() method

`app.project.item(index).layer(index).replaceSource (newSource, fixExpressions)`

Description

Replaces the source for this layer.

Parameters

`newSource`	The new source AVItem object.
`fixExpressions`	`True` to adjust expressions for the new source, `false` otherwise. Note that this feature can be resource-intensive; if replacing a large amount of footage, do this only at the end of the operation. See also "Project autoFixExpressions() method" on page 115.

Returns

Nothing.

AVLayer source attribute

`app.project.item(index).layer(index).source`

Description

The source AVItem for this layer. The value is null in a Text layer. Use `AVLayer.replaceSource()` to change the value.

Type

AVItem object; read-only.

AVLayer sourceRectAtTime() method

app.project.item(index).layer(*index*).sourceRectAtTime(*timeT, extents*)

Description

Retrieves the rectangle bounds of the layer at the specified time index, corrected for text or shape layer content. Use, for example, to write text that is properly aligned to the baseline.

Parameters

timeT	The time index, in seconds. A floating-point value.
extents	True to include the extents, false otherwise. Extents apply to shape layers, increasing the size of the layer bounds as necessary.

Returns

A JavaScript object with four attributes, [top, left, width, height].

AVLayer threeDLayer attribute

app.project.item(index).layer(*index*).threeDLayer

Description

True if this is a 3D layer.

Type

Boolean; read/write.

AVLayer threeDPerChar attribute

app.project.item(index).layer(*index*).threeDPerChar

Description

True if this layer has the Enable Per-character 3D switch set, allowing its characters to be animated off the plane of the text layer. Applies only to text layers.

Type

Boolean; read/write.

AVLayer timeRemapEnabled attribute

app.project.item(index).layer(*index*).timeRemapEnabled

Description

True if time remapping is enabled for this layer.

Type

Boolean; read/write.

AVLayer trackMatteType attribute

`app.project.item(index).layer(`*`index`*`).trackMatteType`

Description

If this layer has a track matte, specifies the way the track matte is applied.

Type

A `TrackMatteType` enumerated value; read/write. One of:

```
TrackMatteType.ALPHA
TrackMatteType.ALPHA_INVERTED
TrackMatteType.LUMA
TrackMatteType.LUMA_INVERTED
TrackMatteType.NO_TRACK_MATTE
```

AVLayer width attribute

`app.project.item(index).layer(`*`index`*`).width`

Description

The width of the layer in pixels.

Type

Floating-point; read-only.

CameraLayer object

`app.project.item(index).layer(index)`

Description

The CameraLayer object represents a camera layer within a composition. Create it using the LayerCollection object's `addCamera` method; see “LayerCollection addCamera() method” on page 96. It can be accessed in an item's layer collection either by index number or by a name string.

- CameraLayer is a subclass of Layer. All methods and attributes of Layer are available when working with CameraLayer. See “Layer object” on page 86.

AE Properties

CameraLayer defines no additional attributes, but has different AE properties than other layer types. It has the following properties and property groups:

```
Marker
Transform
    Point of Interest
    Position
    Scale
    Orientation
    X Rotation
    Y Rotation
    Rotation
    Opacity
Camera Options
    Zoom
    Depth of Field
    Focus Distance
    Blur Level
```

Collection object

Like an array, a collection associates a set of objects or values as a logical group and provides access to them by index. However, most collection objects are read-only. You do not assign objects to them yourself—their contents update automatically as objects are created or deleted.

The index numbering of a collection starts with 1, not 0.

Objects

Object	Reference	Description
`ItemCollection`	"ItemCollection object" on page 82	All of the items (imported files, folders, solids, and so on) found in the Project panel.
`LayerCollection`	"LayerCollection object" on page 95	All of the layers in a composition.
`OMCollection`	"OMCollection object" on page 109	All of the Output Module items in the project.
`RQItemCollection`	"RenderQueueItem object" on page 163	All of the render-queue items in the project.

Attributes

`length`	The number of objects in the collection.

Methods

`[]`	Retrieves an object in the collection by its index number. The first object is at index 1.

CompItem object

```
app.project.item(index)
app.project.items[index]
```

Description

The CompItem object represents a composition, and allows you to manipulate and get information about it. Access the objects by position index number in a project's `item` collection.

- CompItem is a subclass of AVItem, which is a subclass of Item. All methods and attributes of AVItem and Item, in addition to those listed below, are available when working with CompItem. See "AVItem object" on page 30 and "Item object" on page 78.

Example

Given that the first item in the project is a CompItem, the following code displays two alerts. The first shows the number of layers in the CompItem, and the second shows the name of the last layer in the CompItem.

```
var firstComp = app.project.item(1);
alert("number of layers is " + firstComp.numLayers);
alert("name of last layer is " + firstComp.layer(firstComp.numLayers).name);
```

Attributes

Attribute	Reference	Description
`frameDuration`	"CompItem frameDuration attribute" on page 55	The duration of a single frame.
`dropFrame`	"CompItem dropFrame attribute" on page 54	When true, indicates that the composition uses drop-frame timecode.
`workAreaStart`	"CompItem workAreaStart attribute" on page 60	The work area start time.
`workAreaDuration`	"CompItem workAreaDuration attribute" on page 59	The work area duration.
`numLayers`	"CompItem numLayers attribute" on page 57	The number of layers in the composition.
`hideShyLayers`	"CompItem hideShyLayers attribute" on page 55	When true, shy layers are visible in the Timeline panel.
`motionBlur`	"CompItem motionBlur attribute" on page 56	When true, motion blur is enabled for this composition.
`draft3d`	"CompItem draft3d attribute" on page 54	When true, Draft 3D mode is enabled for the Composition panel.
`frameBlending`	"CompItem frameBlending attribute" on page 55	When true, time filtering is enabled for this composition.
`preserveNestedFrameRate`	"CompItem preserveNestedFrameRate attribute" on page 57	When true, the frame rate of nested compositions is preserved.
`preserveNestedResolution`	"CompItem preserveNestedResolution attribute" on page 58	When true, the resolution of nested compositions is preserved.
`bgColor`	"CompItem bgColor attribute" on page 54	The background color of the composition.
`activeCamera`	"CompItem activeCamera attribute" on page 53	The current active camera layer.

Attribute	Reference	Description
displayStartTime	"CompItem displayStartTime attribute" on page 54	Changes the display of the start time in the Timeline panel.
resolutionFactor	"CompItem resolutionFactor attribute" on page 58	The factor by which the *x* and *y* resolution of the Composition panel is downsampled.
shutterAngle	"CompItem shutterAngle attribute" on page 59	The camera shutter angle.
shutterPhase	"CompItem shutterPhase attribute" on page 59	The camera shutter phase.
motionBlurSamplesPerFrame	"CompItem motionBlurSamplesPerFrame attribute" on page 57	The minimum number of motion blur samples per frame for Classic 3D layers, shape layers, and certain effects.
motionBlurAdaptiveSampleLimit	"CompItem motionBlurAdaptiveSample-Limit attribute" on page 56	The maximum number of motion blur samples of 2D layer motion.
layers	"CompItem layers attribute" on page 56 "LayerCollection object" on page 95	The layers of the composition.
selectedLayers	"CompItem selectedLayers attribute" on page 59	The selected layers of the composition.
selectedProperties	"CompItem selectedProperties attribute" on page 59	The selected properties of the composition.
renderer	"CompItem renderer attribute" on page 58	The rendering plug-in module to be used to render this composition.
renderers	"CompItem renderers attribute" on page 58	The set of available rendering plug-in modules.

Methods

Method	Reference	Description
duplicate()	"CompItem duplicate() method" on page 54	Creates and returns a duplicate of this composition.
layer()	"CompItem layer() method" on page 55	Gets a layer from this composition.
openInViewer()	"CompItem openInViewer() method" on page 57	Opens the composition in a Composition panel.

CompItem activeCamera attribute

`app.project.item(index).activeCamera`

Description

The active camera, which is the front-most camera layer that is enabled. The value is null if the composition contains no enabled camera layers.

Type

CameraLayer object; read-only.

CompItem bgColor attribute

`app.project.item(index).bgColor`

Description

The background color of the composition. The three array values specify the red, green, and blue components of the color.

Type

An array containing three floating-point values, [R, G, B], in the range [0.0..1.0]; read/write.

CompItem displayStartTime attribute

`app.project.item(index).displayStartTime`

Description

The time set as the beginning of the composition, in seconds. This is the equivalent of the Start Timecode or Start Frame setting in the Composition Settings dialog box.

Type

Floating-point value in the range [0.0...86339.0] (1 second less than 25 hours); read/write.

CompItem draft3d attribute

`app.project.item(index).draft3d`

Description

When true, Draft 3D mode is enabled for the Composition panel. This corresponds to the value of the Draft 3D button in the Composition panel.

Type

Boolean; read/write.

CompItem dropFrame attribute

`app.project.item(index).dropFrame`

Description

When true, indicates that the composition uses drop-frame timecode. When false, indicates non-drop-frame timecode. This corresponds to the setting in the Composition Settings dialog box.

Type

Boolean; read/write.

CompItem duplicate() method

`app.project.item(index).duplicate()`

Description

Creates and returns a duplicate of this composition, which contains the same layers as the original.

Parameters

None.

Returns

CompItem object.

CompItem frameBlending attribute

`app.project.item(index).frameBlending`

Description

When true, frame blending is enabled for this Composition. Corresponds to the value of the Frame Blending button in the Composition panel.

Type

Boolean; if true, frame blending is enabled; read/write.

CompItem frameDuration attribute

`app.project.item(index).frameDuration`

Description

The duration of a frame, in seconds. This is the inverse of the `frameRate` value (frames-per-second).

Type

Floating-point; read/write.

CompItem hideShyLayers attribute

`app.project.item(index).hideShyLayers`

Description

When true, only layers with `shy` set to false are shown in the Timeline panel. When false, all layers are visible, including those whose `shy` value is true. Corresponds to the value of the Hide All Shy Layers button in the Composition panel.

Type

Boolean; read/write.

CompItem layer() method

```
app.project.item(index).layer(index)
app.project.item(index).layer(otherLayer, relIndex)
app.project.item(index).layer(name)
```

Description

Returns a Layer object, which can be specified by name, an index position in this layer, or an index position relative to another layer.

Parameters

index	The index number of the desired layer in this composition. An integer in the range [1...*numLayers*], where *numLayers* is the number of layers in the composition.

—or—

otherLayer	A Layer object in this composition. The relIndex value is added to the index value of this layer to find the position of the desired layer.
relIndex	The position of the desired layer, relative to otherLayer. An integer in the range [1–otherLayer.index...*numLayers*–otherLayer.index], where *numLayers* is the number of layers in the composition. This value is added to the otherLayer value to derive the absolute index of the layer to return.

—or—

name	The string containing the name of the desired layer.

Returns

Layer object.

CompItem layers attribute

`app.project.item(index).layers`

Description

A LayerCollection object that contains all the Layer objects for layers in this composition. See "LayerCollection object" on page 95.

Type

LayerCollection object; read-only.

CompItem motionBlur attribute

`app.project.item(index).motionBlur`

Description

When true, motion blur is enabled for the composition. Corresponds to the value of the Motion Blur button in the Composition panel.

Type

Boolean; read/write.

CompItem motionBlurAdaptiveSampleLimit attribute

`app.project.item(index).motionBlurAdaptiveSampleLimit`

Description

The maximum number of motion blur samples of 2D layer motion. This corresponds to the Adaptive Sample Limit setting in the Advanced tab of the Composition Settings dialog box.

Type

Integer (between 16 and 256); read/write.

CompItem motionBlurSamplesPerFrame attribute

`app.project.item(index).motionBlurSamplesPerFrame`

Description

The minimum number of motion blur samples per frame for Classic 3D layers, shape layers, and certain effects. This corresponds to the Samples Per Frame setting in the Advanced tab of the Composition Settings dialog box.

Type

Integer (between 2 and 64); read/write.

CompItem numLayers attribute

`app.project.item(index).numLayers`

Description

The number of layers in the composition.

Type

Integer; read-only.

CompItem openInViewer() method

`app.project.item(index).openInViewer()`

Description

Opens the composition in a Composition panel, and moves the Composition panel to front and gives it focus.

Parameters

None.

Returns

Viewer object for the Composition panel, or null if the composition could not be opened.

CompItem preserveNestedFrameRate attribute

`app.project.item(index).preserveNestedFrameRate`

Description

When true, the frame rate of nested compositions is preserved in the current composition. Corresponds to the value of the “Preserve frame rate when nested or in render queue” option in the Advanced tab of the Composition Settings dialog box.

Type

Boolean; read/write.

CompItem preserveNestedResolution attribute

`app.project.item(index).preserveNestedResolution`

Description

When true, the resolution of nested compositions is preserved in the current composition. Corresponds to the value of the "Preserve Resolution When Nested" option in the Advanced tab of the Composition Settings dialog box.

Type

Boolean; read/write.

CompItem renderer attribute

`app.project.item(index).renderer`

Description

The current rendering plug-in module to be used to render this composition, as set in the Advanced tab of the Composition Settings dialog box. Allowed values are the members of *compItem*.renderers.

Type

String; read/write.

CompItem renderers attribute

`app.project.item(index).renderers`

Description

The available rendering plug-in modules. Member strings reflect installed modules, as seen in the Advanced tab of the Composition Settings dialog box.

Type

Array of strings; read-only.

CompItem resolutionFactor attribute

`app.project.item(index).resolutionFactor`

Description

The *x* and *y* downsample resolution factors for rendering the composition.

The two values in the array specify how many pixels to skip when sampling; the first number controls horizontal sampling, the second controls vertical sampling. Full resolution is [1,1], half resolution is [2,2], and quarter resolution is [4,4]. The default is [1,1].

Type

Array of two integers in the range [1..99]; read/write.

CompItem selectedLayers attribute

`app.project.item(index).selectedLayers`

Description

All of the selected layers in this composition. This is a 0-based array (the first object is at index 0).

Type

Array of Layer objects; read-only.

CompItem selectedProperties attribute

`app.project.item(index).selectedProperties`

Description

All of the selected properties (Property and PropertyGroup objects) in this composition. The first property is at index position 0.

Type

Array of Property and PropertyGroup objects; read-only.

CompItem shutterAngle attribute

`app.project.item(index).shutterAngle`

Description

The shutter angle setting for the composition. This corresponds to the Shutter Angle setting in the Advanced tab of the Composition Settings dialog box.

Type

Integer in the range [0...720]; read/write.

CompItem shutterPhase attribute

`app.project.item(index).shutterPhase`

Description

The shutter phase setting for the composition. This corresponds to the Shutter Phase setting in the Advanced tab of the Composition Settings dialog box.

Type

Integer in the range [–360...360]; read/write.

CompItem workAreaDuration attribute

app.project.item*(index)*.workAreaDuration

Description

The duration of the work area in seconds. This is the difference of the start-point and end-point times of the Composition work area.

Type

Floating-point; read/write.

CompItem workAreaStart attribute

`app.project.item(index).workAreaStart`

Description

The time when the Composition work area begins, in seconds.

Type

Floating-point; read/write.

FileSource object

app.project.item(*index*).mainSource
app.project.item(*index*).proxySource

Description

The FileSource object describes footage that comes from a file.

- FileSource is a subclass of FootageSource. All methods and attributes of FootageSource, in addition to those listed below, are available when working with FileSource. See "FootageSource object" on page 69.

Attributes

Attribute	Reference	Description
file	"FileSource file attribute" on page 61	The file that defines this asset.
missingFootagePath	"FileSource missingFootagePath attribute" on page 61	The file that contains footage missing from this asset.

Methods

Method	Reference	Description
reload()	"FileSource reload() method" on page 62	Reloads the asset from the file, if it is a mainSource of a FootageItem.

FileSource file attribute

app.project.item(*index*).mainSource.file
app.project.item(*index*).proxySource.file

Description

The ExtendScript File object for the file that defines this asset. To change the value:

- If this FileSource is a proxySource of an AVItem, call setProxy() or setProxyWithSequence().
- If this FileSource is a mainSource of a FootageItem, call replace() or replaceWithSequence().

Type

File object; read-only.

FileSource missingFootagePath attribute

app.project.item(*index*).mainSource.file.missingFootagePath
app.project.item(*index*).proxySource.file.missingFootagePath

Description

The path and filename of footage that is missing from this asset. See also "AVItem footageMissing attribute" on page 31.

Type

String; read-only.

FileSource reload() method

`app.project.item(`*`index`*`).mainSource.file.mainSource.reload()`

Description

Reloads the asset from the file. This method can be called only on a `mainSource`, not a `proxySource`.

Parameters

None.

Returns

Nothing.

FolderItem object

`app.project.FolderItem`

Description

The FolderItem object corresponds to a folder in your Project panel. It can contain various types of items (footage, compositions, solids) as well as other folders.

Example

Given that the second item in the project is a FolderItem, the following code puts up an alert for each top-level item in the folder, showing that item’s name.

```
var secondItem = app.project.item(2);
if ( !(secondItem instanceof FolderItem) ) {
  alert("problem: second item is not a folder");
} else {
  for (i = 1; i <= secondItem.numItems; i++) {
    alert("item number " + i + " within the folder is named "
      + secondItem.item(i).name);
  }
}
```

Attributes

Attribute	Reference	Description
`items`	“FolderItem items attribute” on page 64	The contents of this folder.
`numItems`	“FolderItem numItems attribute” on page 64	The number of items contained in the folder.

Methods

Method	Reference	Description
`item()`	“FolderItem item() method” on page 63	Gets an item from the folder.

FolderItem item() method

`app.project.item(`*`index`*`).item`

Description

Returns the top-level item in this folder at the specified index position. Note that “top-level” here means top-level within the folder, not necessarily within the project.

Parameters

`index`	An integer, the position index of the item to retrieve. The first item is at index 1.

Returns

Item object.

FolderItem items attribute

`app.project.item(`*`index`*`).items`

Description

An ItemCollection object containing Item object that represent the top-level contents of this folder.

Unlike the ItemCollection in the Project object, this collection contains only the top-level items in the folder. Top-level within the folder is not the same as top-level within the project. Only those items that are top-level in the root folder are also top-level in the Project.

Type

ItemCollection object; read only.

FolderItem numItems attribute

`app.project.item(`*`index`*`).numItems`

Description

The number of items contained in the `items` collection (*`folderItem`*`.items.length`).

If the folder contains another folder, only the FolderItem for that folder is counted, not any subitems contained in it.

Type

Integer; read only.

FootageItem object

`app.project.item(`*`index`*`)`
`app.project.items[`*`index`*`]`

Description

The FootageItem object represents a footage item imported into a project, which appears in the Project panel. These are accessed by position index number in a project's `item` collection.

- FootageItem is a subclass of AVItem, which is a subclass of Item. All methods and attributes of AVItem and Item, in addition to those listed below, are available when working with FootageItem. See "AVItem object" on page 30 and "Item object" on page 78.

Attributes

Attribute	Reference	Description
`file`	"FootageItem file attribute" on page 65	The footage source file.
`mainSource`	"FootageItem mainSource attribute" on page 66	All settings related to the footage item.

Methods

Method	Reference	Description
`replace()`	"FootageItem replace() method" on page 66	Replaces a footage file with another footage file.
`replaceWithPlaceholder()`	"FootageItem replaceWithPlaceholder() method" on page 67	Replaces a footage file with a placeholder object.
`replaceWithSequence()`	"FootageItem replaceWithSequence() method" on page 67	Replaces a footage file with an image sequence.
`replaceWithSolid()`	"FootageItem replaceWithSolid() method" on page 67	Replaces a footage file with a solid.
`openInViewer()`	"FootageItem openInViewer() method" on page 66	Opens the footage in a Footage panel.

FootageItem file attribute

`app.project.item(`*`index`*`).file`

Description

The ExtendScript File object for the footage's source file.

If the FootageItem's `mainSource` is a FileSource, this is the same as *`FootageItem`*`.mainSource.file`. Otherwise it is null.

Type

File object; read only.

FootageItem mainSource attribute

`app.project.item(`*`index`*`).mainSource`

Description

The footage source, an object that contains all of the settings related to that footage item, including those that are normally accessed through the Interpret Footage dialog box. The attribute is read-only. To change its value, call one of the FootageItem "replace" methods.

See the "FootageSource object" on page 69, and its three types:

- "SolidSource object" on page 179
- "FileSource object" on page 61
- "PlaceholderSource object" on page 113

If this is a FileSource object, and the `footageMissing` value is true, the path to the missing footage file is in the `FileSource.missingFootagePath` attribute. See "AVItem footageMissing attribute" on page 31 and "FileSource missingFootagePath attribute" on page 61.

Type

FootageSource object; read-only.

FootageItem openInViewer() method

`app.project.item(index).openInViewer()`

Description

Opens the footage in a Footage panel, and moves the Footage panel to front and gives it focus.

NOTE: Missing and placeholder footage can be opened using this method, but cannot manually (via double-clicking it).

Parameters

None.

Returns

Viewer object for the Footage panel, or null if the footage could not be opened.

FootageItem replace() method

`app.project.item(`*`index`*`).replace(`*`file`*`)`

Description

Changes the source of this FootageItem to the specified file. In addition to loading the file, the method creates a new FileSource object for the file and sets `mainSource` to that object. In the new source object, it sets the `name`, `width`, `height`, `frameDuration`, and `duration` attributes (see "AVItem object" on page 30) based on the contents of the file.

The method preserves interpretation parameters from the previous `mainSource` object. If the specified file has an unlabeled alpha channel, the method estimates the alpha interpretation.

Parameters

`file`	An ExtendScript File object for the file to be used as the footage main source.

FootageItem replaceWithPlaceholder() method

`app.project.item(`*`index`*`).replaceWithPlaceholder(`*`name, width, height, frameRate, duration`*`)`

Description

Changes the source of this FootageItem to the specified placeholder. Creates a new PlaceholderSource object, sets its values from the parameters, and sets `mainSource` to that object.

Parameters

`name`	A string containing the name of the placeholder.
`width`	The width of the placeholder in pixels, an integer in the range [4..30000].
`height`	The height of the placeholder in pixels, an integer in the range [4..30000].
`frameRate`	The frame rate of the placeholder, a floating-point value in the range [1.0..99.0]
`duration`	The duration of the placeholder in seconds, a floating-point value in the range [0.0..10800.0].

FootageItem replaceWithSequence() method

`app.project.item(`*`index`*`).replaceWithSequence(`*`file, forceAlphabetical`*`)`

Description

Changes the source of this FootageItem to the specified image sequence. In addition to loading the file, the method creates a new FileSource object for the file and sets `mainSource` to that object. In the new source object, it sets the `name`, `width`, `height`, `frameDuration`, and `duration` attributes (see "AVItem object" on page 30) based on the contents of the file.

The method preserves interpretation parameters from the previous `mainSource` object. If the specified file has an unlabeled alpha channel, the method estimates the alpha interpretation.

Parameters

`file`	An ExtendScript File object for the first file in the sequence to be used as the footage main source.
`forceAlphabetical`	When true, use the "Force alphabetical order" option.

FootageItem replaceWithSolid() method

`app.project.item(`*`index`*`).replaceWithSolid(`*`color, name, width, height, pixelAspect`*`)`

Description

Changes the source of this FootageItem to the specified solid. Creates a new SolidSource object, sets its values from the parameters, and sets `mainSource` to that object.

Parameters

`color`	The color of the solid, an array of three floating-point values, [R, G, B], in the range [0.0..1.0].
`name`	A string containing the name of the solid.
`width`	The width of the solid in pixels, an integer in the range [4..30000].
`height`	The height of the solid in pixels, an integer in the range [4..30000].
`pixelAspect`	The pixel aspect ratio of the solid, a floating-point value in the range [0.01..100.0].

FootageSource object

`app.project.item(index).mainSource`
`app.project.item(index).proxySource`

Description

The FootageSource object holds information describing the source of some footage. It is used as the `mainSource` of a FootageItem, or the `proxySource` of a CompItem or FootageItem. See “FootageItem object” on page 65 and “CompItem object” on page 52.

- FootageSource is the base class for SolidSource, so FootageSource attributes and methods are available when working with SolidSource objects. See “SolidSource object” on page 179.

Attributes

Attribute	Reference	Description
`hasAlpha`	“FootageSource hasAlpha attribute” on page 72	When true, a footage clip or proxy includes an alpha channel.
`alphaMode`	“FootageSource alphaMode attribute” on page 70	The mode of an alpha channel.
`premulColor`	“FootageSource premulColor attribute” on page 73	The color to be premultiplied.
`invertAlpha`	“FootageSource invertAlpha attribute” on page 72	When true, an alpha channel in a footage clip or proxy should be inverted.
`isStill`	“FootageSource isStill attribute” on page 72	When true, footage is a still image.
`fieldSeparationType`	“FootageSource fieldSeparationType attribute” on page 71	The field separation type.
`highQualityFieldSeparation`	“FootageSource highQualityFieldSeparation attribute” on page 72	How the fields are to be separated in non-still footage.
`removePulldown`	“FootageSource removePulldown attribute” on page 73	The pulldown type for the footage.
`loop`	“FootageSource loop attribute” on page 73	How many times an image sequence is set to loop.
`nativeFrameRate`	“FootageSource nativeFrameRate attribute” on page 73	The native frame rate of the footage.
`displayFrameRate`	“FootageSource displayFrameRate attribute” on page 70	The effective frame rate as displayed and rendered in compositions by After Effects.
`conformFrameRate`	“FootageSource conformFrameRate attribute” on page 70	The rate to which footage should conform.

Methods

Method	Reference	Description
`guessAlphaMode()`	“FootageSource guessAlphaMode() method” on page 71	Estimates the `alphaMode` setting.
`guessPulldown()`	“FootageSource guessPulldown() method” on page 71	Estimates the `pulldownType` setting.

FootageSource alphaMode attribute

app.project.item*(index)*.mainSource.alphaMode
app.project.item*(index)*.proxySource.alphaMode

Description

The alphaMode attribute of footageSource defines how the alpha information in the footage is to be interpreted. If hasAlpha is false, this attribute has no relevant meaning.

Type

An AlphaMode enumerated value; (read/write). One of:

AlphaMode.IGNORE
AlphaMode.STRAIGHT
AlphaMode.PREMULTIPLIED

FootageSource conformFrameRate attribute

app.project.item*(index)*.mainSource.conformFrameRate
app.project.item*(index)*.proxySource.conformFrameRate

Description

A frame rate to use instead of the nativeFrameRate value. If set to 0, the nativeFrameRate is used instead.

It is an error to set this value if FootageSource.isStill is true. It is an error to set this value to 0 if removePulldown is not set to PulldownPhase.OFF. If this is 0 when you set removePulldown to a value other than PulldownPhase.OFF, then this is automatically set to the value of nativeFrameRate.

Type

Floating-point value in the range [0.0.. 99.0]; read/write.

FootageSource displayFrameRate attribute

app.project.item*(index)*.mainSource.displayFrameRate
app.project.item*(index)*.proxySource.displayFrameRate

Description

The effective frame rate as displayed and rendered in compositions by After Effects.

If removePulldown is PulldownPhase.OFF, then this is the same as the conformFrameRate (if non-zero) or the nativeFrameRate (if conformFrameRate is 0). If removePulldown is not PulldownPhase.OFF, this is conformFrameRate * 0.8, the effective frame rate after removing 1 of every 5 frames.

Type

Floating-point value in the range [0.0.. 99.0]; read-only.

FootageSource fieldSeparationType attribute

app.project.item*(index)*.mainSource.fieldSeparationType
app.project.item*(index)*.proxySource.fieldSeparationType

Description

How the fields are to be separated in non-still footage.

It is an error to set this attribute if isStill is true. It is an error to set this value to FieldSeparationType.OFF if removePulldown is not PulldownPhase.OFF.

Type

A FieldSeparationType enumerated value; read/write. One of:

FieldSeparationType.OFF
FieldSeparationType.UPPER_FIELD_FIRST
FieldSeparationType.LOWER_FIELD_FIRST

FootageSource guessAlphaMode() method

app.project.item*(index)*.mainSource.guessAlphaMode()
app.project.item*(index)*.proxySource.guessAlphaMode()

Description

Sets alphaMode, premulColor, and invertAlpha to the best estimates for this footage source. If hasAlpha is false, no change is made.

Parameters

None.

Returns

Nothing.

FootageSource guessPulldown() method

app.project.item*(index)*.mainSource.guessPulldown(*method*)
app.project.item*(index)*.proxySource.guessPulldown(*method*)

Description

Sets fieldSeparationType and removePulldown to the best estimates for this footage source. If isStill is true, no change is made.

Parameters

method	The method to use for estimation. A PulldownMethod enumerated value, one of: PulldownMethod.PULLDOWN_3_2 PulldownMethod.ADVANCE_24P

Returns

Nothing.

FootageSource hasAlpha attribute

app.project.item*(index)*.mainSource.hasAlpha
app.project.item*(index)*.proxySource.hasAlpha

Description

When true, the footage has an alpha component. In this case, the attributes alphaMode, invertAlpha, and premulColor have valid values. When false, those attributes have no relevant meaning for the footage.

Type

Boolean; read-only.

FootageSource highQualityFieldSeparation attribute

app.project.item*(index)*.mainSource.highQualityFieldSeparation
app.project.item*(index)*.proxySource.highQualityFieldSeparation

Description

When true, After Effects uses special algorithms to determine how to perform high-quality field separation.

It is an error to set this attribute if isStill is true, or if fieldSeparationType is FieldSeparationType.OFF.

Type

Boolean; read/write.

FootageSource invertAlpha attribute

app.project.item*(index)*.mainSource.invertAlpha
app.project.item*(index)*.proxySource.invertAlpha

Description

When true, an alpha channel in a footage clip or proxy should be inverted.

This attribute is valid only if an alpha is present. If hasAlpha is false, or if alphaMode is AlphaMode.IGNORE, this attribute is ignored.

Type

Boolean; read/write.

FootageSource isStill attribute

app.project.item*(index)*.mainSource.isStill
app.project.item*(index)*.proxySource.isStill

Description

When true the footage is still; when false, it has a time-based component.

Examples of still footage are JPEG files, solids, and placeholders with duration of 0. Examples of non-still footage are movie files, sound files, sequences, and placeholders of non-zero duration.

Type

Boolean; read-only.

FootageSource loop attribute

app.project.item*(index)*.mainSource.loop
app.project.item*(index)*.proxySource.loop

Description

The number of times that the footage is to be played consecutively when used in a composition.

It is an error to set this attribute if isStill is true.

Type

Integer in the range [1.. 9999]; default is 1; read/write.

FootageSource nativeFrameRate attribute

app.project.item*(index)*.mainSource.nativeFrameRate
app.project.item*(index)*.proxySource.nativeFrameRate

Description

The native frame rate of the footage.

Type

Floating-point; read/write.

FootageSource premulColor attribute

app.project.item*(index)*.mainSource.premulColor
app.project.item*(index)*.proxySource.premulColor

Description

The color to be premultiplied. This attribute is valid only if the alphaMode is alphaMode.PREMULTIPLIED.

Type

Array of three floating-point values [R, G, B], in the range [0.0..1.0]; read/write.

FootageSource removePulldown attribute

app.project.item*(index)*.mainSource.removePulldown
app.project.item*(index)*.proxySource.removePulldown

Description

How the pulldowns are to be removed when field separation is used.

It is an error to set this attribute if isStill is true. It is an error to attempt to set this to a value other than PulldownPhase.OFF in the case where fieldSeparationType is FieldSeparationType.OFF.

Type

A PulldownPhase enumerated value; read/write. One of:

PulldownPhase.RemovePulldown.OFF
PulldownPhase.RemovePulldown.WSSWW
PulldownPhase.RemovePulldown.SSWWW

```
PulldownPhase.RemovePulldown.SWWWS
PulldownPhase.RemovePulldown.WWWSS
PulldownPhase.RemovePulldown.WWSSW
PulldownPhase.RemovePulldown.WSSWW_24P_ADVANCE
PulldownPhase.RemovePulldown.SSWWW_24P_ADVANCE
PulldownPhase.RemovePulldown.SWWWS_24P_ADVANCE
PulldownPhase.RemovePulldown.WWWSS_24P_ADVANCE
PulldownPhase.RemovePulldown.WWSSW_24P_ADVANCE
```

ImportOptions object

```
new ImportOptions();
new ImportOptions(file);
```

Description

The ImportOptions object encapsulates the options used to import a file with the `Project.importFile` methods. See "Project importFile() method" on page 118.

The constructor takes an optional parameter, an ExtendScript File object for the file. If it is not supplied, you must explicitly set the value of the `file` attribute before using the object with the `importFile` method. For example:

```
new ImportOptions().file = new File("myfile.psd");
```

Attributes

Attributes	Reference	Description
`importAs`	"ImportOptions importAs attribute" on page 76	The type of file to be imported.
`sequence`	"ImportOptions sequence attribute" on page 77	When true, import a sequence of files, rather than an individual file.
`forceAlphabetical`	"ImportOptions forceAlphabetical attribute" on page 76	When true, the "Force alphabetical order" option is set.
`file`	"ImportOptions file attribute" on page 76	The file to import, or the first file of the sequence to import.

Methods

Method	Reference	Description
`canImportAs()`	"ImportOptions canImportAs() method" on page 75	Restricts input to a particular file type.

ImportOptions canImportAs() method

`importOptions``.canImportAs(`*`type`*`)`

Description

Reports whether the file can be imported as the source of a particular object type. If this method returns true, you can set the given type as the value of the `importAs` attribute. See "ImportOptions importAs attribute" on page 76.

Parameters

`type`	The type of file that can be imported. An `ImportAsType` enumerated value; one of:
	`ImportAsType.COMP`
	`ImportAsType.FOOTAGE`
	`ImportAsType.COMP_CROPPED_LAYERS`
	`ImportAsType.PROJECT`

Returns

Boolean.

Example

```
var io = new ImportOptions(File("c:\\myFile.psd"));
if io.canImportAs(ImportAsType.COMP);
  io.importAs = ImportAsType.COMP;
```

ImportOptions file attribute

importOptions.file

Description

The file to be imported. If a file is set in the constructor, you can access it through this attribute.

Type

ExtendScript File object; read/write.

ImportOptions forceAlphabetical attribute

importOptions.forceAlphabetical

Description

When true, has the same effect as setting the "Force alphabetical order" option in the File > Import > File dialog box.

Type

Boolean; read/write.

ImportOptions importAs attribute

importOptions.importAs

Description

The type of object for which the imported file is to be the source. Before setting, use `canImportAs` to check that a given file can be imported as the source of the given object type. See "ImportOptions canImportAs() method" on page 75.

Type

An `ImportAsType` enumerated value; read/write. One of:

```
ImportAsType.COMP_CROPPED_LAYERS
ImportAsType.FOOTAGE
ImportAsType.COMP
ImportAsType.PROJECT
```

ImportOptions sequence attribute

importOptions.sequence

Description

When true, a sequence is imported; otherwise, an individual file is imported.

Type

Boolean; read/write.

Item object

```
app.project.item(index)
app.project.items[index]
```

Description

The Item object represents an item that can appear in the Project panel.

The first item is at index 1.

- Item is the base class for AVItem and for FolderItem, which are in turn the base classes for various other item types, so Item attributes and methods are available when working with all of these item types. See "AVItem object" on page 30 and "FolderItem object" on page 63.

Attributes

Attributes	Reference	Description
`name`	"Item name attribute" on page 79	The name of the object as shown in the Project panel.
`comment`	"Item comment attribute" on page 79	A descriptive string.
`id`	"Item id attribute" on page 79	A unique identifier for this item.
`parentFolder`	"Item parentFolder attribute" on page 80	The parent folder of this item.
`selected`	"Item selected attribute" on page 80	When true, this item is currently selected.
`typeName`	"Item typeName attribute" on page 81	The type of item.
`label`	"Item label attribute" on page 79	The label color for the item.

Methods

Method	Reference	Description
`remove()`	"Item remove() method" on page 80	Deletes the item from the project.

Example

This example gets the second item from the project and checks that it is a folder. It then removes from the folder any top-level item that is not currently selected. It also checks to make sure that, for each item in the folder, the parent is properly set to the correct folder.

```
var myFolder = app.project.item(2);
if (myFolder.typeName != "Folder") {
  alert("error: second item is not a folder");
}
else {
  var numInFolder = myFolder.numItems;
  // Always run loops backwards when deleting things:
  for(i = numInFolder; i >= 1; i--) {
    var curItem = myFolder.item(i);
    if ( curItem.parentFolder != myFolder) {
      alert("error within AE: the parentFolder is not set correctly");
    }
    else {
      if ( !curItem.selected && curItem.typeName == "Footage") {
          //found an unselected solid.
          curItem.remove();
        }
      }
    }
  }
```

Item comment attribute

`app.project.item(`*`index`*`).comment`

Description

A string that holds a comment, up to 15,999 bytes in length after any encoding conversion. The comment is for the user's purpose only; it has no effect on the item's appearance or behavior.

Type

String; read/write.

Item id attribute

`app.project.item(`*`index`*`).id`

Description

A unique and persistent identification number used internally to identify an item between sessions. The value of the ID remains the same when the project is saved to a file and later reloaded. However, when you import this project into another project, new IDs are assigned to all items in the imported project. The ID is not displayed anywhere in the user interface.

Type

Integer; read-only.

Item label attribute

`app.project.item(index).label`

Description

The label color for the item. Colors are represented by their number (0 for None, or 1 to 16 for one of the preset colors in the Labels preferences).

Custom label colors cannot be set programmatically.

Type

Integer (0 to 16); read/write.

Item name attribute

`app.project.item(index).name`

Description

The name of the item as displayed in the Project panel.

Type

String; read/write.

Item parentFolder attribute

`app.project.item(index).parentFolder`

Description

The FolderItem object for the folder that contains this item. If this item is at the top level of the project, this is the project's root folder (`app.project.rootFolder`). You can use the ItemCollection's `addFolder` method to add a new folder, and set this value to put items in the new folder. See "ItemCollection addFolder() method" on page 82.

Type

FolderItem object; read/write.

Example

This script creates a new FolderItem in the Project panel and moves compositions into it.

```
// create a new FolderItem in project, with name "comps"
var compFolder = app.project.items.addFolder("comps");
// move all compositions into new folder by setting
// compItem's parentFolder to "comps" folder
for(var i = 1; i <= app.project.numItems; i++) {
  if(app.project.item(i) instanceof CompItem)
  app.project.item(i).parentFolder = compFolder;
}
```

Item remove() method

`app.project.item(index).remove()`

Description

Deletes this item from the project and from the Project panel. If the item is a FolderItem, all the items contained in the folder are also removed from the project. No files or folders are removed from disk.

Parameters

None.

Returns

Nothing.

Item selected attribute

`app.project.item(index).selected`

Description

When true, this item is selected. Multiple items can be selected at the same time. Set to true to select the item programmatically, or to false to deselect it.

Type

Boolean; read/write.

Item typeName attribute

`app.project.item(`*`index`*`).typeName`

Description

A user-readable name for the item type; for example, “Folder”, “Footage”, or “Composition”.

Type

String; read-only.

ItemCollection object

`app.project.items`

Description

The ItemCollection object represents a collection of items. The ItemCollection belonging to a Project object contains all the Item objects for items in the project. The ItemCollection belonging to a FolderItem object contains all the Item objects for items in that folder.

- ItemCollection is a subclass of Collection. All methods and attributes of Collection, in addition to those listed below, are available when working with ItemCollection. See "Collection object" on page 51.

Methods

Method	Reference	Description
`addComp()`	"ItemCollection addComp() method" on page 82	Creates a new CompItem object and adds it to the collection.
`addFolder()`	"ItemCollection addFolder() method" on page 82	Creates a new FolderItem object and adds it to the collection.

ItemCollection addComp() method

`app.project.items.addComp(`*`name, width, height, pixelAspect, duration, frameRate`*`)`

Description

Creates a new composition. Creates and returns a new CompItem object and adds it to this collection.

If the ItemCollection belongs to the project or the root folder, then the new item's `parentFolder` is the root folder. If the ItemCollection belongs to any other folder, the new item's `parentFolder` is that `FolderItem`.

Parameters

`name`	A string containing the name of the composition.
`width`	The width of the composition in pixels, an integer in the range [4..30000].
`height`	The height of the composition in pixels, an integer in the range [4..30000].
`pixelAspect`	The pixel aspect ratio of the composition, a floating-point value in the range [0.01..100.0].
`duration`	The duration of the composition in seconds, a floating-point value in the range [0.0..10800.0].
`frameRate`	The frame rate of the composition, a floating-point value in the range [1.0..99.0]

Returns

CompItem object.

ItemCollection addFolder() method

`app.project.items.addFolder(`*`name`*`)`

Description

Creates a new folder. Creates and returns a new FolderItem object and adds it to this collection.

If the ItemCollection belongs to the project or the root folder, then the new folder's `parentFolder` is the root folder. If the ItemCollection belongs to any other folder, the new folder's `parentFolder` is that `FolderItem`.

To put items in the folder, set the item object's `parentFolder` attribute; see "Item parentFolder attribute" on page 80.

Parameters

`name`	A string containing the name of the folder.

Returns

FolderItem object.

Example

This script creates a new FolderItem in the Project panel and moves compositions into it.

```
// create a new FolderItem in project, with name "comps"
var compFolder = app.project.items.addFolder("comps");
// move all compositions into new folder by setting
// compItem's parentFolder to "comps" folder
for(var i = 1; i <= app.project.numItems; i++) {
  if(app.project.item(i) instanceof CompItem)
  app.project.item(i).parentFolder = compFolder;
}
```

KeyframeEase object

`myKey = new KeyframeEase(speed, influence);`

Description

The KeyframeEase object encapsulates the keyframe ease settings of a layer's AE property. Keyframe ease is determined by the speed and influence values that you set using the property's `setTemporalEaseAtKey` method. See "Property setTemporalEaseAtKey() method" on page 144.

The constructor creates a KeyframeEase object. Both parameters are required.

- `speed`: A floating-point value. Sets the `speed` attribute.
- `influence`: A floating-point value in the range [0.1..100.0]. Sets the `influence` attribute.

Example

This example assumes that the Position, a spatial property, has more than two keyframes.

```
var easeIn = new KeyframeEase(0.5, 50);
var easeOut = new KeyframeEase(0.75, 85);
var myPositionProperty = app.project.item(1).layer(1).property("Position")
myPositionProperty.setTemporalEaseAtKey(2, [easeIn], [easeOut]);
```

This example sets the Scale, a temporal property with either two or three dimensions. For 2D and 3D properties you must set an `easeIn` and `easeOut` value for each dimension:

```
var easeIn = new KeyframeEase(0.5, 50);
var easeOut = new KeyframeEase(0.75, 85);
var myScaleProperty = app.project.item(1).layer(1).property("Scale")
myScaleProperty.setTemporalEaseAtKey(2, [easeIn, easeIn, easeIn], [easeOut, easeOut, easeOut]);
```

Attributes

Attribute	Reference	Description
`speed`	"KeyframeEase speed attribute" on page 85	The speed setting for a keyframe.
`influence`	"KeyframeEase influence attribute" on page 84	The influence setting for a keyframe.

KeyframeEase influence attribute

myKey.`influence`

Description

The influence value of the keyframe, as shown in the Keyframe Velocity dialog box.

Type

Floating-point value in the range [0.1..100.0]; read/write.

KeyframeEase speed attribute

myKey.speed

Description

The speed value of the keyframe. The units depend on the type of keyframe, and are displayed in the Keyframe Velocity dialog box.

Type

Floating-point value; read/write.

Layer object

`app.project.item(index).layer(index)`

Description

The Layer object provides access to layers within compositions. It can be accessed from an item’s layer collection either by index number or by a name string.

- Layer is the base class for CameraLayer, LightLayer, and AVLayer, so Layer attributes and methods are available when working with all layer types. See “AVLayer object” on page 38, “CameraLayer object” on page 50, and “LightLayer object” on page 100.

Layers contain AE properties, in addition to their JavaScript attributes and methods. For examples of how to access properties in layers, see “PropertyBase object” on page 148.

Example

If the first item in the project is a CompItem, this example disables the first layer in that composition and renames it. This might, for example, turn an icon off in the composition.

```
var firstLayer = app.project.item(1).layer(1);
firstLayer.enabled = false;
firstLayer.name = "Disabled Layer";
```

Attributes

Attribute	Reference	Description
`index`	“Layer index attribute” on page 90	The index position of the layer.
`name`	“Layer name attribute” on page 92	The name of the layer.
`parent`	“Layer parent attribute” on page 92	The parent of this layer.
`time`	“Layer time attribute” on page 94	The current time of the layer.
`startTime`	“Layer startTime attribute” on page 94	The start time of the layer.
`stretch`	“Layer stretch attribute” on page 94	The time stretch percentage of the layer.
`inPoint`	“Layer inPoint attribute” on page 90	The “in” point of the layer.
`outPoint`	“Layer outPoint attribute” on page 92	The “out” point of the layer.
`enabled`	“Layer enabled attribute” on page 89	When true, the layer is enabled.
`solo`	“Layer solo attribute” on page 94	When true, the layer is soloed.
`shy`	“Layer shy attribute” on page 93	When true, the layer is shy.
`locked`	“Layer locked attribute” on page 90	When true, the layer is locked.
`hasVideo`	“Layer hasVideo attribute” on page 89	When true, the layer contains a video component.
`active`	“Layer active attribute” on page 87	When true, the layer is active at the current time.
`nullLayer`	“Layer nullLayer attribute” on page 92	When true, this is a null layer.
`selectedProperties`	“Layer selectedProperties attribute” on page 93	All selected AE properties in the layer.
`comment`	“Layer comment attribute” on page 88	A descriptive comment for the layer.
`containingComp`	“Layer containingComp attribute” on page 88	The composition that contains this layer.

Attribute	Reference	Description
isNameSet	"Layer isNameSet attribute" on page 90	When true, the layer's name has been explicitly set.

Methods

Method	Reference	Description
remove()	"Layer remove() method" on page 93	Deletes the layer from the composition.
moveToBeginning()	"Layer moveToBeginning() method" on page 91	Moves the layer to the top of the composition (makes it the first layer).
moveToEnd()	"Layer moveToEnd() method" on page 91	Moves the layer to the bottom of the composition (makes it the last layer).
moveAfter()	"Layer moveAfter() method" on page 90	Moves the layer below another layer.
moveBefore()	"Layer moveBefore() method" on page 91	Moves the layer above another layer.
duplicate()	"Layer duplicate() method" on page 89	Duplicates the layer.
copyToComp()	"Layer copyToComp() method" on page 89	Copies the layer to the top (beginning) of another composition.
activeAtTime()	"Layer activeAtTime() method" on page 87	Reports whether this layer will be active at a specified time.
setParentWithJump()	"Layer setParentWithJump() method" on page 93	Sets a new parent for this layer.
applyPreset()	"Layer applyPreset() method" on page 88	Applies a named collection of animation settings to the layer.

Layer active attribute

`app.project.item(index).layer(index).active`

Description

When true, the layer's video is active at the current time.

For this to be true, the layer must be enabled, no other layer may be soloing unless this layer is soloed too, and the time must be between the `inPoint` and `outPoint` values of this layer.

This value is never true in an audio layer; there is a separate `audioActive` attribute in the AVLayer object.

Type

Boolean; read-only.

Layer activeAtTime() method

app.project.item(index).layer(index).activeAtTime(*time*)

Description

Returns true if this layer will be active at the specified time. To return true, the layer must be enabled, no other layer may be soloing unless this layer is soloed too, and the time must be between the `inPoint` and `outPoint` values of this layer.

Parameters

time	The time in seconds, a floating-point value.

Returns

Boolean.

Layer applyPreset() method

appapp.project.item(index).layer(index).applyPreset(*presetName*);

Description

Applies the specified collection of animation settings (an animation preset) to the layer. Predefined animation preset files are installed in the Presets folder, and users can create new animation presets through the user interface.

Parameters

presetName	An ExtendScript File object for the file containing the animation preset.

Returns

Nothing.

Layer comment attribute

app.project.item(index).layer(index).comment

Description

A descriptive comment for the layer.

Type

String; read/write.

Layer containingComp attribute

app.project.item(index).layer(index).containingComp

Description

The composition that contains this layer.

Type

CompItem object; read-only.

Layer copyToComp() method

`app.project.item(index).layer(index).copyToComp(`*`intoComp`*`)`

Description

Copies the layer into the specified composition. The original layer remains unchanged. Creates a new Layer object with the same values as this one, and prepends the new object to the `layers` collection in the target CompItem. Retrieve the copy using *`intoComp`*`.layer(1)`.

Copying in a layer changes the index positions of previously existing layers in the target composition. This is the same as copying and pasting a layer through the user interface.

Parameters

`intoComp`	The target composition, and CompItem object.

Returns

Nothing.

Layer duplicate() method

`app.project.item(index).layer(index).duplicate()`

Description

Duplicates the layer. Creates a new Layer object in which all values are the same as in this one. This has the same effect as selecting a layer in the user interface and choosing Edit > Duplicate, except the selection in the user interface does not change when you call this method.

Parameters

None.

Returns

Layer object.

Layer enabled attribute

`app.project.item(index).layer(index).enabled`

Description

When true, the layer is enabled; otherwise false. This corresponds to the video switch state of the layer in the Timeline panel.

Type

Boolean; read/write.

Layer hasVideo attribute

`app.project.item(index).layer(index).hasVideo`

Description

When true, the layer has a video switch (the eyeball icon) in the Timeline panel; otherwise false.

Type

Boolean; read-only.

Layer index attribute

`app.project.item(index).layer(index).index`

Description

The index position of the layer.

Type

Integer in the range [1..`numLayers`]; read-only.

Layer inPoint attribute

`app.project.item(index).layer(index).inPoint`

Description

The “in” point of the layer, expressed in composition time (seconds).

Type

Floating-point value in the range [-10800.0..10800.0] (minus or plus three hours); read/write.

Layer isNameSet attribute

`app.project.item(index).layer(index).isNameSet`

Description

True if the value of the `name` attribute has been set explicitly, rather than automatically from the source.

Type

Boolean; read-only.

Layer locked attribute

`app.project.item(index).layer(index).locked`

Description

When true, the layer is locked; otherwise false. This corresponds to the lock toggle in the Layer panel.

Type

Boolean; read/write.

Layer moveAfter() method

`app.project.item(index).layer(index).moveAfter(`*`layer`*`)`

Description

Moves this layer to a position immediately after (below) the specified layer.

Parameters

`layer`	The target layer, a layer object in the same composition.

Returns

Nothing.

Layer moveBefore() method

`app.project.item(index).layer(index).moveBefore(`*`layer`*`)`

Description

Moves this layer to a position immediately before (above) the specified layer.

Parameters

`layer`	The target layer, a layer object in the same composition.

Returns

Nothing.

Layer moveToBeginning() method

`app.project.item(index).layer(index).moveToBeginning()`

Description

Moves this layer to the topmost position of the layer stack (the first layer).

Parameters

None.

Returns

Nothing.

Layer moveToEnd() method

`app.project.item(index).layer(index).moveToEnd()`

Description

Moves this layer to the bottom position of the layer stack (the last layer).

Parameters

None.

Returns

Nothing.

Layer name attribute

`app.project.item(index).layer(index).name`

Description

The name of the layer. By default, this is the same as the Source name (which cannot be changed in the Layer panel), but you can set it to be different.

Type

String; read/write.

Layer nullLayer attribute

`app.project.item(index).layer(index).nullLayer`

Description

When true, the layer was created as a null object; otherwise false.

Type

Boolean; read-only.

Layer outPoint attribute

`app.project.item(index).layer(index).outPoint`

Description

The “out” point of the layer, expressed in composition time (seconds).

Type

Floating-point value in the range [-10800.0..10800.0] (minus or plus three hours); read/write.

Layer parent attribute

`app.project.item(index).layer(index).parent`

Description

The parent of this layer; can be null.

Offset values are calculated to counterbalance any transforms above this layer in the hierarchy, so that when you set the parent there is no apparent jump in the layer's transform. For example, if the new parent has a rotation of 30 degrees, the child layer is assigned a rotation of -30 degrees.

To set the parent without changing the child layer's transform values, use the `setParentWithJump` method.

Type

Layer object or null; read/write.

Layer remove() method

`app.project.item(index).layer(index).remove()`

Description

Deletes the specified layer from the composition.

Parameters

None.

Returns

Nothing.

Layer selectedProperties attribute

`app.project.item(index).layer(index).selectedProperties`

Description

An array containing all of the currently selected Property and PropertyGroup objects in the layer.

Type

Array of PropertyBase objects; read-only.

Layer setParentWithJump() method

`app.project.item(index).layer(index).setParentWithJump()`
`app.project.item(index).layer(index).setParentWithJump(`*`newParent`*`)`

Description

Sets the parent of this layer to the specified layer, without changing the transform values of the child layer. There may be an apparent jump in the rotation, translation, or scale of the child layer, as this layer's transform values are combined with those of its ancestors.

If you do not want the child layer to jump, set the `parent` attribute directly. In this case, an offset is calculated and set in the child layer's transform fields, to prevent the jump from occurring.

Parameters

`newParent`	Optional, a layer object in the same composition. If not specified, it sets the parent to None.

Returns

Nothing.

Layer shy attribute

`app.project.item(index).layer(index).shy`

Description

When true, the layer is "shy," meaning that it is hidden in the Layer panel if the composition's "Hide all shy layers" option is toggled on.

Type

Boolean; read/write.

Layer solo attribute

`app.project.item(index).layer(index).solo`

Description

When true, the layer is soloed, otherwise false.

Type

Boolean; read/write.

Layer startTime attribute

`app.project.item(index).layer(index).startTime`

Description

The start time of the layer, expressed in composition time (seconds).

Type

Floating-point value in the range [-10800.0..10800.0] (minus or plus three hours); read/write.

Layer stretch attribute

`app.project.item(index).layer(index).stretch`

Description

The layer’s time stretch, expressed as a percentage. A value of 100 means no stretch. Values between 0 and 1 are set to 1, and values between -1 and 0 (not including 0) are set to -1.

Type

Floating-point value in the range [-9900.0..9900.0]; read/write.

Layer time attribute

`app.project.item(index).layer(index).time`

Description

The current time of the layer, expressed in composition time (seconds).

Type

Floating-point value; read-only.

LayerCollection object

`app.project.item(index).layers`

Description

The LayerCollection object represents a set of layers. The LayerCollection belonging to a CompItem object contains all the layer objects for layers in the composition. The methods of the collection object allow you to manipulate the layer list.

- LayerCollection is a subclass of Collection. All methods and attributes of Collection, in addition to those listed below, are available when working with LayerCollection. See "Collection object" on page 51.

Example

Given that the first item in the project is a CompItem and the second item is an AVItem, this example shows the number of layers in the CompItem's layer collection, adds a new layer based on an AVItem in the project, then displays the new number of layers.

```
var firstComp = app.project.item(1);
var layerCollection = firstComp.layers;
alert("number of layers before is " + layerCollection.length);
var anAVItem = app.project.item(2);
layerCollection.add(anAVItem);
alert("number of layers after is " + layerCollection.length);
```

Methods

Method	Reference	Description
`add()`	"LayerCollection add() method" on page 96	Creates a new AVLayer and adds it to this collection.
`addNull()`	"LayerCollection addNull() method" on page 97	Creates a new, null layer and adds it to this collection.
`addSolid()`	"LayerCollection addSolid() method" on page 98	Creates a new layer, a FootageItem with a SolidSource, and adds it to this collection.
`addText()`	"LayerCollection addText() method" on page 98	Creates a new point text layer and adds it to this collection.
`addBoxText()`	"LayerCollection addBoxText() method" on page 96	Creates a new paragraph (box) text layer and adds it to this collection.
`addCamera()`	"LayerCollection addCamera() method" on page 96	Creates a new camera layer and adds it to this collection.
`addLight()`	"LayerCollection addLight() method" on page 97	Creates a new light layer and adds it to this collection.
`addShape()`	"LayerCollection addShape() method" on page 97	Creates a new shape layer and adds it to this collection.
`byName()`	"LayerCollection byName() method" on page 99	Retrieves the layer object with a specified name.
`precompose()`	"LayerCollection precompose() method" on page 99	Collects specified layers into a new composition.

LayerCollection add() method

`app.project.item(index).layers.add(`*`item, duration`*`)`

Description

Creates a new AVLayer object containing the specified item, and adds it to this collection.

The new layer honors the Create Layers at Composition Start Time preference.

This method generates an exception if the item cannot be added as a layer to this CompItem.

Parameters

`item`	The AVItem object for the item to be added.
`duration`	Optional, the length of a still layer in seconds, a floating-point value. Used only if the item contains a piece of still footage. Has no effect on movies, sequences or audio. If supplied, sets the `duration` value of the new layer. Otherwise, the `duration` value is set according to user preferences. By default, this is the same as the duration of the containing CompItem. To set another preferred value, choose Edit > Preferences > Import (Windows) or After Effects > Preferences > Import (Mac OS), and specify options under Still Footage.

Returns

AVLayer object.

LayerCollection addBoxText() method

`app.project.item(index).layers.addBoxText(`*`sourceText`*`)`

Description

Creates a new paragraph (box) text layer and adds the new TextLayer object to this collection.

To create a point text layer, use the `addText()` method. For more information, see “LayerCollection addText() method” on page 98.

Parameters

`sourceText`	Optional; a string containing the source text of the new layer, or a TextDocument object containing the source text of the new layer. See “TextDocument object” on page 182.

Returns

TextLayer object.

LayerCollection addCamera() method

`app.project.item(index).layers.addCamera(`*`name, centerPoint`*`)`

Description

Creates a new camera layer and adds the CameraLayer object to this collection.

The new layer honors the Create Layers at Composition Start Time preference.

Parameters

`name`	A string containing the name of the new layer.

centerPoint	The center of the new camera, a floating-point array [x, y]. This is used to set the initial x and y values of the new camera's Point of Interest property. The z value is set to 0.

Returns

CameraLayer object.

LayerCollection addLight() method

app.project.item(index).layers.addLight(*name, centerPoint*)

Description

Creates a new light layer and adds the LightLayer object to this collection.

The new layer honors the Create Layers at Composition Start Time preference.

Parameters

name	A string containing the name of the new layer.
centerPoint	The center of the new light, a floating-point array [x, y].

Returns

LightLayer object.

LayerCollection addNull() method

app.project.item(index).layers.addNull(*duration*)

Description

Creates a new null layer and adds the AVLayer object to this collection. This is the same as choosing Layer > New > Null Object.

Parameters

duration	Optional, the length of a still layer in seconds, a floating-point value. If supplied, sets the duration value of the new layer. Otherwise, the duration value is set according to user preferences. By default, this is the same as the duration of the containing CompItem. To set another preferred value, choose Edit > Preferences > Import (Windows) or After Effects > Preferences > Import (Mac OS), and specify options under Still Footage.

Returns

AVLayer object.

LayerCollection addShape() method

app.project.item(index).layers.addShape()

Description

Creates a new ShapeLayer object for a new, empty Shape layer. Use the ShapeLayer object to add properties, such as shape, fill, stroke, and path filters.

This is the same as using a shape tool in "Tool Creates Shape" mode. Tools automatically add a vector group that includes Fill and Stroke as specified in the tool options.

Parameters

None.

Returns

ShapeLayer object.

LayerCollection addSolid() method

`app.project.item(index).layers.addSolid(`*`color, name, width, height, pixelAspect, duration`*`)`

Description

Creates a new SolidSource object, with values set as specified; sets the new SolidSource as the `mainSource` value of a new FootageItem object, and adds the FootageItem to the project. Creates a new AVLayer object, sets the new FootageItem as its `source`, and adds the layer to this collection.

Parameters

`color`	The color of the solid, an array of three floating-point values, [R, G, B], in the range [0.0..1.0].
`name`	A string containing the name of the solid.
`width`	The width of the solid in pixels, an integer in the range [4..30000].
`height`	The height of the solid in pixels, an integer in the range [4..30000].
`pixelAspect`	The pixel aspect ratio of the solid, a floating-point value in the range [0.01..100.0].
`duration`	Optional, the length of a still layer in seconds, a floating-point value. If supplied, sets the `duration` value of the new layer. Otherwise, the `duration` value is set according to user preferences. By default, this is the same as the duration of the containing CompItem. To set another preferred value, choose Edit > Preferences > Import (Windows) or After Effects > Preferences > Import (Mac OS), and specify options under Still Footage.

Returns

AVLayer object.

LayerCollection addText() method

`app.project.item(index).layers.addText(`*`sourceText`*`)`

Description

Creates a new point text layer and adds the new TextLayer object to this collection.

To create a paragraph (box) text layer, use the `addBoxText()` method. For more information, see "LayerCollection addBoxText() method" on page 96.

Parameters

`sourceText`	Optional; a string containing the source text of the new layer, or a TextDocument object containing the source text of the new layer. See "TextDocument object" on page 182.

Returns

TextLayer object.

LayerCollection byName() method

`app.project.item(index).layers.byName(`*`name`*`)`

Description

Returns the first (topmost) layer found in this collection with the specified name, or null if no layer with the given name is found.

Parameters

`name`	A string containing the name.

Returns

Layer object or null.

LayerCollection precompose() method

`app.project.item(index).layers.precompose(`*`layerIndicies, name, moveAllAttributes`*`)`

Description

Creates a new CompItem object and moves the specified layers into its layer collection. It removes the individual layers from this collection, and adds the new CompItem to this collection.

Parameters

`layerIndices`	The position indexes of the layers to be collected. An array of integers.
`name`	The name of the new CompItem object.
`moveAllAttributes`	Optional. When true (the default), retains all attributes in the new composition. This is the same as selecting the “Move all attributes into the new composition” option in the Pre-compose dialog box. You can only set this to false if there is just one index in the `layerIndices` array. This is the same as selecting the “Leave all attributes in” option in the Pre-compose dialog box.

Returns

CompItem object.

LightLayer object

`app.project.item(index).layer(index)`

Description

The LightLayer object represents a light layer within a composition. Create it using the LayerCollection object's `addLight` method; see "LayerCollection addLight() method" on page 97. It can be accessed in an item's layer collection either by index number or by a name string.

- LightLayer is a subclass of Layer. All methods and attributes of Layer are available when working with Light-Layer. See "Layer object" on page 86.

AE Properties

LightLayer defines no additional attributes, but has different AE properties than other layer types. It has the following properties and property groups:

```
Marker
Transform
    Point of Interest
    Position
    Scale
    Orientation
    X Rotation
    Y Rotation
    Rotation
    Opacity
Light Options
    Intensity
    Color
    Cone Angle
    Cone Feather
    Casts Shadows
    Shadow Darkness
    Shadow Diffusion
```

Attributes

Attribute	Reference	Description
`lightType`	"LightLayer lightType attribute" on page 100	For light layers, the type of light.

LightLayer lightType attribute

`app.project.item(index).layer(index).lightType`

Description

For a light layer, its light type.

Trying to set this attribute for a non-light layer produces an error.

Type

A LightType enumerated value; read/write. One of:

LightType.PARALLEL
LightType.SPOT
LightType.POINT
LightType.AMBIENT

MarkerValue object

new MarkerValue(*comment, chapter, url, frameTarget, cuePointName, params*)

Description

The MarkerValue object represents a layer marker, which associates a comment, and optionally a chapter reference point, Web-page link, or Flash Video cue point with a particular point in a layer. Create it with the constructor; all arguments except comment are optional. All arguments are strings that set in the corresponding attributes of the returned MarkerValue object, except params. This is an array containing key-value pairs., which can then be accessed with the getParameters() and setParameters() methods. A script can set any number of parameter pairs; the order does not reflect the order displayed in the application.

To associate a marker with a layer, set the MarkerValue object in the Marker AE property of the layer:

layerObject.property("Marker").setValueAtTime(*time*, *markerValueObject*);
For information on the usage of markers see “Using markers” in After Effects Help.

Attributes

Attribute	Reference	Description
comment	“MarkerValue comment attribute” on page 103	A comment on the associated layer.
duration	“MarkerValue duration attribute” on page 103	The amount of time represented by the marker.
chapter	“MarkerValue chapter attribute” on page 103	A chapter link reference point for the associated layer.
cuePointName	“MarkerValue cuePointName attribute” on page 103	The Flash Video cue point name.
eventCuePoint	“MarkerValue eventCuePoint attribute” on page 104	Whether the Flash Video cue point is for an event or navigation.
url	“MarkerValue url attribute” on page 105	A URL for Web page to be associated with the layer.
frameTarget	“MarkerValue frameTarget attribute” on page 104	A specific frame target within the Web page specified by url.

Methods

Method	Reference	Description
getParameters()	“MarkerValue getParameters() method” on page 104	Retrieves the key-value pairs associated with the marker value.
setParameters()	“MarkerValue setParameters() method” on page 104	Sets the key-value pairs associated with the marker value.

Examples

- To set a marker that says “Fade Up” at the 2 second mark:

```
var myMarker = new MarkerValue("Fade Up");
myLayer.property("Marker").setValueAtTime(2, myMarker);
```

- To get comment values from a particular marker:

```
var commentOfFirstMarker = app.project.item(1).layer(1).property("Marker").keyValue(1).comment;
var commentOfMarkerAtTime4  =
    app.project.item(1).layer(1).property("Marker").valueAtTime(4.0,true).comment
var markerProperty = app.project.item(1).layer(1).property("Marker");
var markerValueAtTimeClosestToTime4 =
    markerProperty.keyValue(markerProperty.nearestKeyIndex(4.0));
var commentOfMarkerClosestToTime4 = markerValueAtTimeClosestToTime4.comment;
```

MarkerValue chapter attribute

app.project.item(*index*).layer(*index*).property("Marker").keyValue(*index*).chapter

Description

A text chapter link for this marker. Chapter links initiate a jump to a chapter in a QuickTime movie or in other formats that support chapter marks.

Type

String; read/write.

MarkerValue comment attribute

app.project.item(*index*).layer(*index*).property("Marker").keyValue(*index*).comment

Description

A text comment for this marker. This comment appears in the Timeline panel next to the layer marker.

Type

String; read/write.

MarkerValue cuePointName attribute

app.project.item(*index*).layer(*index*).property("Marker").keyValue(*index*).cuePointName

Description

The Flash Video cue point name, as shown in the Marker dialog box.

Type

String; read/write.

MarkerValue duration attribute

app.project.item(*index*).layer(*index*).property("Marker").keyValue(*index*).duration

Description

The marker’s duration, in seconds. The duration appears in the Timeline panel as a short bar extending from the marker location.

Type

Floating point; read/write.

MarkerValue eventCuePoint attribute

`app.project.item(index).layer(index).property("Marker").keyValue(index).eventCuePoint`

Description

When `true`, the FlashVideo cue point is for an event; otherwise, it is for navigation.

Type

Boolean; read/write.

MarkerValue frameTarget attribute

`app.project.item(index).layer(index).property("Marker").keyValue(index).frameTarget`

Description

A text frame target for this marker. Together with the URL value, this targets a specific frame within a Web page.

Type

String; read/write.

MarkerValue getParameters() method

`app.project.item(index).layer(index).property("Marker").keyValue(index).getParameters()`

Description

Returns the key-value pairs for Flash Video cue-point parameters, for a cue point associated with this marker value.

Parameters

None.

Returns

An object with an attribute matching each parameter name, containing that parameter’s value.

MarkerValue setParameters() method

`app.project.item(index).layer(index).property("Marker").keyValue(index).setParameters(keyValuePairs)`

Description

Associates a set of key-value pairs for Flash Video cue-point parameters, for a cue point associated with this marker value. A cue point can have any number of parameters, but you can add only three through the user interface; use this method to add more than three parameters.

Parameters

`keyValuePairs`	An object containing the key-value pairs as attributes and values. The object’s `toString()` method is called to assign the string value of each attribute to the named key.

Returns

Nothing.

Example

```
var mv = new MarkerValue("My Marker");

var parms = new Object;
parms.timeToBlink = 1;
parms.assignMe = "A string"

mv.setParameters(parms);

myLayer.property("Marker").setValueAtTime(2, mv);
```

MarkerValue url attribute

app.project.item(*index*).layer(*index*).property("Marker").keyValue(*index*).url

Description

A URL for this marker. This URL is an automatic link to a Web page.

Type

String; read/write.

MaskPropertyGroup object

`app.project.item(index).layer(index).mask`

Description

The MaskPropertyGroup object encapsulates mask attributes in a layer.

- MaskPropertyGroup is a subclass of PropertyGroup. All methods and attributes of PropertyBase and PropertyGroup, in addition to those listed below, are available when working with MaskPropertyGroup. See "PropertyBase object" on page 148 and "PropertyGroup object" on page 155.

Attributes

Attribute	Reference	Description
`maskMode`	"MaskPropertyGroup maskMode attribute" on page 107	The mask mode.
`inverted`	"MaskPropertyGroup inverted attribute" on page 106	When true, the mask is inverted.
`rotoBezier`	"MaskPropertyGroup rotoBezier attribute" on page 108	When true, the shape of the mask is RotoBezier.
`maskMotionBlur`	"MaskPropertyGroup maskMotionBlur attribute" on page 107	How motion blur is applied to this mask.
`locked`	"MaskPropertyGroup locked attribute" on page 107	When true, the mask is locked.
`color`	"MaskPropertyGroup color attribute" on page 106	The color used to draw the mask outline in the user interface.
`maskFeatherFalloff`	"MaskPropertyGroup maskFeatherFalloff attribute" on page 107	The feather falloff mode for the mask.

MaskPropertyGroup color attribute

`app.project.item(index).layer(index).mask(`*`index`*`).color`

Description

The color used to draw the mask outline as it appears in the user interface (Composition panel, Layer panel, and Timeline panel).

Type

Array of three floating-point values, [R, G, B], in the range [0.0..1.0]; read/write.

MaskPropertyGroup inverted attribute

`app.project.item(index).layer(index).mask(`*`index`*`).inverted`

Description

When true, the mask is inverted; otherwise false.

Type

Boolean; read/write.

MaskPropertyGroup locked attribute

`app.project.item(index).layer(index).mask(`*`index`*`).locked`

Description

When true, the mask is locked and cannot be edited in the user interface; otherwise, false.

Type

Boolean; read/write.

MaskPropertyGroup maskFeatherFalloff attribute

`app.project.item(index).layer(index).mask(`*`index`*`).maskFeatherFalloff`

Description

The feather falloff mode for the mask. Equivalent to the Layer > Mask > Feather Falloff setting.

Type

A `MaskFeatherFalloff` enumerated value; read/write. One of:

```
MaskFeatherFalloff.FFO_LINEAR
MaskFeatherFalloff.FFO_SMOOTH
```

MaskPropertyGroup maskMode attribute

`app.project.item(index).layer(index).mask(`*`index`*`).maskMode`

Description

The masking mode for this mask.

Type

A `MaskMode` enumerated value; read/write. One of:

```
MaskMode.NONE
MaskMode.ADD
MaskMode.SUBTRACT
MaskMode.INTERSECT
MaskMode.LIGHTEN
MaskMode.DARKEN
MaskMode.DIFFERENCE
```

MaskPropertyGroup maskMotionBlur attribute

`app.project.item(index).layer(index).mask(`*`index`*`).maskMotionBlur`

Description

How motion blur is applied to this mask.

Type

A `MakMotionBlur` enumerated value; read/write. One of:

```
MaskMotionBlur.SAME_AS_LAYER
MaskMotionBlur.ON
MaskMotionBlur.OFF
```

MaskPropertyGroup rotoBezier attribute

`app.project.item(index).layer(index).`*`mask(index)`*`.rotoBezier`

Description

When true, the mask is a RotoBezier shape; otherwise, false.

Type

Boolean; read/write.

OMCollection object

`app.project.renderQueue.items.outputModules`

Description

The OMCollection contains all of the output modules in a render queue. The collection provides access to the OutputModule objects, but does not provide any additional functionality. The first OutputModule object in the collection is at index position 1. See “OutputModule object” on page 110

- OMCollection is a subclass of Collection. All methods and attributes of Collection are available when working with OMCollection. See “Collection object” on page 51.

OutputModule object

app.project.renderQueue.item*(index)*.outputModule(*index*)

Description

An OutputModule object of a RenderQueueItem generates a single file or sequence via a render operation, and contains attributes and methods relating to the file to be rendered.

Attributes

Attribute	Reference	Description
file	"OutputModule file attribute" on page 111	The path and name of the file to be rendered.
postRenderAction	"OutputModule postRenderAction attribute" on page 111	An action to be taken after rendering.
name	"OutputModule name attribute" on page 111	The user-interface name of the output module.
templates	"OutputModule templates attribute" on page 112	All templates for the output module.
includeSourceXMP	"OutputModule includeSourceXMP attribute" on page 111	When true, writes all source footage XMP metadata to the output file.

Methods

Method	Reference	Description
remove()	"OutputModule remove() method" on page 112	Removes this output module from the render-queue item's list.
saveAsTemplate()	"OutputModule saveAsTemplate() method" on page 112	Saves a new output-module template.
applyTemplate()	"OutputModule applyTemplate() method" on page 110	Applies an output-module template.

OutputModule applyTemplate() method

app.project.renderQueue.item*(index)*.outputModule(*index*).applyTemplate(*templateName*)

Description

Applies the specified existing output-module template.

Parameters

templateName	A string containing the name of the template to be applied.

Returns

Nothing.

OutputModule file attribute

`app.project.renderQueue.item(index).outputModule(index).file`

Description

The ExtendScript File object for the file this output module is set to render.

Type

ExtendScript File object; read/write.

OutputModule includeSourceXMP attribute

`app.project.renderQueue.item(index).outputModule(index).includeSourceXMP`

Description

When true, writes all source footage XMP metadata to the output file. Corresponds to the Include Source XMP Metadata option in the Output Module Settings dialog box.

Type

Boolean; read/write.

OutputModule name attribute

`app.project.renderQueue.item(index).outputModule(index).name`

Description

The name of the output module, as shown in the user interface.

Type

String; read-only.

OutputModule postRenderAction attribute

`app.project.renderQueue.item(index).outputModule(index).postRenderAction`

Description

An action to be performed when the render operation is completed.

Type

A `PostRenderAction` enumerated value (read/write); one of:

```
postRenderAction.NONE
postRenderAction.IMPORT
postRenderAction.IMPORT_AND_REPLACE_USAGE
postRenderAction.SET_PROXY
```

OutputModule remove() method

app.project.renderQueue.item*(index)*.outputModule(*index*).remove()

Description

Removes this OutputModule object from the collection.

Parameters

None.

Returns

Nothing.

OutputModule saveAsTemplate() method

app.project.renderQueue.item*(index)*.outputModule(*index*).saveAsTemplate(*name*)

Description

Saves this output module as a template and adds it to the templates array.

Parameters

name	A string containing the name of the new template.

Returns

Nothing.

OutputModule templates attribute

app.project.renderQueue.item*(index)*.outputModule(*index*).templates

Description

The names of all output-module templates available in the local installation of After Effects.

Type

Array of strings; read-only.

PlaceholderSource object

`app.project.item(`*`index`*`).mainSource`
`app.project.item(`*`index`*`).proxySource`

Description

The PlaceholderSource object describes the footage source of a placeholder.

PlaceholderSource is a subclass of FootageSource. All methods and attributes of FootageSource are available when working with PlaceholderSource. See “FootageSource object” on page 69. PlaceholderSource does not define any additional methods or attributes.

Project object

`app.project`

Description

The project object represents an After Effects project. Attributes provide access to specific objects within the project, such as imported files or footage and compositions, and also to project settings such as the timecode base. Methods can import footage, create solids, compositions and folders, and save changes.

Attributes

Attribute	Reference	Description
`file`	"Project file attribute" on page 117	The file for the currently open project.
`rootFolder`	"Project rootFolder attribute" on page 121	The folder containing all the contents of the project; the equivalent of the Project panel
`items`	"Project items attribute" on page 119	All items in the project.
`activeItem`	"Project activeItem attribute" on page 115	The currently active item.
`bitsPerChannel`	"Project bitsPerChannel attribute" on page 116	The color depth of the current project.
`transparencyGridThumbnails`	"Project transparencyGridThumbnails attribute" on page 123	When true, thumbnail views use the transparency checkerboard pattern.
`numItems`	"Project numItems attribute" on page 120	The total number of items contained in the project.
`selection`	"Project selection attribute" on page 122	All items selected in the Project panel.
`renderQueue`	"Project renderQueue attribute" on page 121	The project's render queue.
`timeDisplayType`	"Project timeDisplayType attribute" on page 122	The time display style, corresponding to the Time Display Style section in the Project Settings dialog box.
`footageTimecodeDisplay-StartType`	"Project footageTimecodeDisplayStart-Type attribute" on page 117	The Footage Start Time setting in the Project Settings dialog box, which is enabled when Timecode is selected as the time display style.
`framesUseFeetFrames`	"Project framesUseFeetFrames attribute" on page 118	The Use Feet + Frames setting in the Project Settings dialog box.
`feetFramesFilmType`	"Project feetFramesFilmType attribute" on page 117	The Use Feet + Frames menu setting in the Project Settings dialog box.
`framesCountType`	"Project framesCountType attribute" on page 118	The Frame Count menu setting in the Project Settings dialog box.
`displayStartFrame`	"Project displayStartFrame attribute" on page 117	The frame at which to start numbering when displaying the project.
`linearBlending`	"Project linearBlending attribute" on page 120	When true, linear blending is used for the project.
`xmpPacket`	"Project xmpPacket attribute" on page 123	The project's XMP metadata.

Methods

Method	Reference	Description
`item()`	"Project item() method" on page 119	Retrieves an item from the project.
`consolidateFootage()`	"Project consolidateFootage() method" on page 116	Consolidates all footage in the project.
`removeUnusedFootage()`	"Project removeUnusedFootage() method" on page 121	Removes unused footage from the project.
`reduceProject()`	"Project reduceProject() method" on page 120	Reduces the project to a specified set of items.
`close()`	"Project close() method" on page 116	Closes the project with normal save options.
`save()`	"Project save() method" on page 121	Saves the project.
`saveWithDialog()`	"Project saveWithDialog() method" on page 122	Displays a Save dialog box.
`importPlaceholder()`	"Project importFileWithDialog() method" on page 119	Imports a placeholder into the project.
`importFile()`	"Project importFile() method" on page 118	Imports a file into the project.
`importFileWithDialog()`	"Project importFileWithDialog() method" on page 119	Displays an Import File dialog box.
`showWindow()`	"Project showWindow() method" on page 122	Shows or hides the Project panel.
`autoFixExpressions()`	"Project autoFixExpressions() method" on page 115	Automatically replaces text in all expressions.

Project activeItem attribute

`app.project.activeItem`

Description

The item that is currently active and is to be acted upon, or a null if no item is currently selected or if multiple items are selected.

Type

Item object or null; read-only.

Project autoFixExpressions() method

`app.project.autoFixExpressions(`*`oldText, newText`*`)`

Description

Automatically replaces text found in broken expressions in the project, if the new text causes the expression to evaluate without errors.

Parameters

`oldText`	The text to replace.
`newText`	The new text.

Returns

Nothing.

Project bitsPerChannel attribute

`app.project.bitsPerChannel`

Description

The color depth of the current project, either 8, 16, or 32 bits.

Type

Integer (8, 16, or 32 only); read/write.

Project close() method

`app.project.close(`*`closeOptions`*`)`

Description

Closes the project with the option of saving changes automatically, prompting the user to save changes or closing without saving changes.

Parameters

`closeOptions`	Action to be performed on close. A `CloseOptions` enumerated value, one of: `CloseOptions.DO_NOT_SAVE_CHANGES`: Close without saving. `CloseOptions.PROMPT_TO_SAVE_CHANGES`: Prompt for whether to save changes before close. `CloseOptions.SAVE_CHANGES`: Save automatically on close.

Returns

Boolean. True on success. False if the file has not been previously saved, the user is prompted, and the user cancels the save.

Project consolidateFootage() method

`app.project.consolidateFootage()`

Description

Consolidates all footage in the project. Same as the File > Consolidate All Footage command.

Parameters

None.

Returns

Integer; the total number of footage items removed.

Project displayStartFrame attribute

`app.project.displayStartFrame`

Description

An alternate way of setting the Frame Count menu setting in the Project Settings dialog box to 0 or 1, and is equivalent to using the `FramesCountType.FC_START_0` or `FramesCountType.FC_START_1` enumerated values for the `framesCountType` attribute. For more information, see “Project framesCountType attribute” on page 118.

Type

Integer (0 or 1); read/write.

Project feetFramesFilmType attribute

`app.project.feetFramesFilmType`

Description

The Use Feet + Frames menu setting in the Project Settings dialog box.

Use this attribute instead of the old `timecodeFilmType` attribute.

Type

A `FeetFramesFilmType` enumerated value; read/write. One of:

```
FeetFramesFilmType.MM16
FeetFramesFilmType.MM35
```

Project file attribute

`app.project.file`

Description

The ExtendScript File object for the file containing the project that is currently open.

Type

File object or null if project has not been saved; read-only.

Project footageTimecodeDisplayStartType attribute

`app.project.footageTimecodeDisplayStartType`

Description

The Footage Start Time setting in the Project Settings dialog box, which is enabled when Timecode is selected as the time display style.

Type

A `FootageTimecodeDisplayStartType` enumerated value; read/write. One of:

```
FootageTimecodeDisplayStartType.FTCS_START_0
FootageTimecodeDisplayStartType.FTCS_USE_SOURCE_MEDIA
```

Project framesCountType attribute

`app.project.framesCountType`

Description

The Frame Count menu setting in the Project Settings dialog box.

Type

A `FramesCountType` enumerated value; read/write. One of:

```
FramesCountType.FC_START_1
FramesCountType.FC_START_0
FramesCountType.FC_TIMECODE_CONVERSION
```

NOTE: Setting this attribute to `FramesCountType.FC_TIMECODE_CONVERSION` *resets the* `displayStart-Frame` *attribute to 0.*

Project framesUseFeetFrames attribute

`app.project.framesUseFeetFrames`

Description

The Use Feet + Frames setting in the Project Settings dialog box. True if using Feet + Frames; false if using Frames.

Type

Boolean; read/write.

Project importFile() method

`app.project.importFile(`*`importOptions`*`)`

Description

Imports the file specified in the specified ImportOptions object, using the specified options. Same as the File > Import File command. Creates and returns a new FootageItem object from the file, and adds it to the project’s `items` array.

Parameters

`importOptions`	An ImportOptions object specifying the file to import and the options for the operation. See “ImportOptions object” on page 75.

Returns

FootageItem object.

Example

```
app.project.importFile(new ImportOptions(File(“sample.psd”))
```

Project importFileWithDialog() method

`app.project.importFileWithDialog()`

Description

Shows an Import File dialog box. Same as the File > Import > File command.

Returns

Array of Item objects created during import; or null if the user cancels the dialog box.

Project importPlaceholder() method

`app.project.importPlaceholder(`*`name, width, height, frameRate, duration`*`)`

Description

Creates and returns a new PlaceholderItem object and adds it to the project's `items` array. Same as the File > Import > Placeholder command.

Parameters

`name`	A string containing the name of the placeholder.
`width`	The width of the placeholder in pixels, an integer in the range [4..30000].
`height`	The height of the placeholder in pixels, an integer in the range [4..30000].
`frameRate`	The frame rate of the placeholder, a floating-point value in the range [1.0..99.0]
`duration`	The duration of the placeholder in seconds, a floating-point value in the range [0.0..10800.0].

Returns

PlaceholderItem object.

Project item() method

`app.project.item(`*`index`*`)`

Description

Retrieves an item at a specified index position.

Parameters

`index`	The index position of the item, an integer. The first item is at index 1.

Returns

Item object.

Project items attribute

`app.project.items`

Description

All of the items in the project.

Type

ItemCollection object; read-only.

Project linearBlending attribute

`app.project.linearBlending`

Description

True if linear blending should be used for this project; otherwise false.

Type

Boolean; read/write.

Project numItems attribute

`app.project.numItems`

Description

The total number of items contained in the project, including folders and all types of footage.

Type

Integer; read-only.

Example

```
n = app.project.numItems;
alert("There are " + n + " items in this project.")
```

Project reduceProject() method

`app.project.reduceProject(`*`array_of_items`*`)`

Description

Removes all items from the project except those specified. Same as the File > Reduce Project command.

Parameters

`array_of_items`	An array containing the Item objects that are to be kept.

Returns

Integer; the total number of items removed.

Example

```
var theItems = new Array();
theItems[theItems.length] = app.project.item(1);
theItems[theItems.length] = app.project.item(3);
app.project.reduceProject(theItems);
```

Project removeUnusedFootage() method

`app.project.removeUnusedFootage()`

Description

Removes unused footage from the project. Same as the File > Remove Unused Footage command.

Parameters

None.

Returns

Integer; the total number of FootageItem objects removed.

Project renderQueue attribute

`app.project.renderQueue`

Description

The render queue of the project.

Type

RenderQueue object; read-only.

Project rootFolder attribute

`app.project.rootFolder`

Description

The root folder containing the contents of the project; this is a virtual folder that contains all items in the Project panel, but not items contained inside other folders in the Project panel.

Type

FolderItem object; read-only.

Project save() method

`app.project.save()`
`app.project.save(`*`file`*`)`

Description

Saves the project. The same as the File > Save or File > Save As command. If the project has never previously been saved and no file is specified, prompts the user for a location and file name. Pass a File object to save a project to a new file without prompting.

Parameters

`file`	Optional. An ExtendScript File object for the file to save.

Returns

None.

Project saveWithDialog() method

`app.project.saveWithDialog()`

Description

Shows the Save dialog box. The user can name a file with a location and save the project, or click Cancel to exit the dialog box.

Parameters

None.

Returns

Boolean; true if the project was saved.

Project selection attribute

`app.project.selection`

Description

All items selected in the Project panel, in the sort order shown in the Project panel.

Type

Array of Item objects; read-only.

Project showWindow() method

app.project.showWindow(*doShow*)

Description

Shows or hides the Project panel.

Parameters

`doShow`	When true, show the Project panel. When false, hide the Project panel.

Returns

Nothing.

Project timeDisplayType attribute

`app.project.timeDisplayType`

Description

The time display style, corresponding to the Time Display Style section in the Project Settings dialog box.

Type

A `TimeDisplayType` enumerated value; read/write. One of:

```
TimeDisplayType.FRAMES
TimeDisplayType.TIMECODE
```

Project transparencyGridThumbnails attribute

`app.project.transparencyGridThumbnails`

Description

When true, thumbnail views use the transparency checkerboard pattern.

Type

Boolean; read/write.

Project xmpPacket attribute

`app.project.xmpPacket`

Description

The project's XMP metadata, stored as RDF (XML-based). For more information on XMP, see the *JavaScript Tools Guide*.

Type

String; read/write.

Example

The following example code accesses the XMP metadata of the current project, and modifies the Label project metadata field.

```
var proj = app.project;

// load the XMP library as an ExtendScript ExternalObject
if (ExternalObject.AdobeXMPScript == undefined) {
  ExternalObject.AdobeXMPScript = new
  ExternalObject('lib:AdobeXMPScript');
}

var mdata = new XMPMeta(app.project.xmpPacket);  // get the project's XMP metadata

// update the Label project metadata's value
var schemaNS = XMPMeta.getNamespaceURI("xmp");
var propName = "xmp:Label";
try {
  mdata.setProperty(schemaNS, propName, "final version...no, really!");
}
catch(e) {
  alert(e);
}
app.project.xmpPacket = mdata.serialize();
```

Property object

app.project.item(*index*).layer(*index*).*propertySpec*

Description

The Property object contains value, keyframe, and expression information about a particular AE property of a layer. An AE property is an value, often animatable, of an effect, mask, or transform within an individual layer. For examples of how to access properties, see “PropertyBase object” on page 148 and “PropertyGroup property() method” on page 157.

- Property is a subclass of PropertyBase. All methods and attributes of PropertyBase, in addition to those listed below, are available when working with Property. See “PropertyBase object” on page 148.

NOTE: JavaScript objects commonly referred to as “properties” are called “attributes” in this guide, to avoid confusion with the After Effects definition of property.

Attributes

Attribute	Reference	Description
propertyValueType	“Property propertyValueType attribute” on page 138	Type of value stored in this property.
value	“Property value attribute” on page 146	Current value of the property.
hasMin	“Property hasMin attribute” on page 130	When true, there is a minimum permitted value.
hasMax	“Property hasMax attribute” on page 130	When true, there is a maximum permitted value.
minValue	“Property minValue attribute” on page 137	The minimum permitted value.
maxValue	“Property maxValue attribute” on page 137	The maximum permitted value.
isSpatial	“Property isSpatial attribute” on page 132	When true, the property defines a spatial value.
canVaryOverTime	“Property canVaryOverTime attribute” on page 129	When true, the property can be keyframed.
isTimeVarying	“Property isTimeVarying attribute” on page 132	When true, the property has keyframes or an expression enabled that can vary its values.
numKeys	“Property numKeys attribute” on page 138	The number of keyframes on this property.
unitsText	“Property unitsText attribute” on page 146	A text description of the units in which the value is expressed.
expression	“Property expression attribute” on page 129	The expression string for this property.
canSetExpression	“Property canSetExpression attribute” on page 128	When true, the expression can be set by a script.
expressionEnabled	“Property expressionEnabled attribute” on page 129	When true, the expression is used to generate values for the property.
expressionError	“Property expressionError attribute” on page 130	The error, if any, that occurred when the last expression was evaluated.

Attribute	Reference	Description
selectedKeys	"Property selectedKeys attribute" on page 140	All selected keyframes of the property.
propertyIndex	"Property propertyIndex attribute" on page 138	The position index of this property.
dimensionsSeparated	"Property dimensionsSeparated attribute" on page 129	When true, the property's dimensions are represented as separate properties.
isSeparationFollower	"Property isSeparationFollower attribute" on page 131	When true, the property represents one of the separated dimensions for a multidimensional property.
isSeparationLeader	"Property isSeparationLeader attribute" on page 131	When true, the property is multidimensional and can be separated.
separationDimension	"Property separationDimension attribute" on page 140	For a separated follower, the dimension it represents in the multidimensional leader.
separationLeader	"Property separationLeader attribute" on page 140	The original multidimensional property for this separated follower.

Methods

Method	Reference	Description
valueAtTime()	"Property valueAtTime() method" on page 146	Gets the value of the property evaluated at given time.
setValue()	"Property setValue() method" on page 144	Sets the static value of the property.
setValueAtTime()	"Property setValueAtTime() method" on page 145	Creates a keyframe for the property.
setValuesAtTimes()	"Property setValuesAtTimes() method" on page 145	Creates a set of keyframes for the property.
setValueAtKey()	"Property setValueAtKey() method" on page 145	Finds a keyframe and sets the value of the property at that keyframe.
nearestKeyIndex()	"Property nearestKeyIndex() method" on page 138	Gets the keyframe nearest to a specified time.
keyTime()	"Property keyTime() method" on page 136	Gets the time at which a condition occurs.
keyValue()	"Property keyValue() method" on page 137	Gets the value of a keyframe at the time at which a condition occurs.
addKey()	"Property addKey() method" on page 128	Adds a new keyframe to the property at a given time.
removeKey()	"Property removeKey() method" on page 139	Removes a keyframe from the property.
isInterpolationTypeValid()	"Property isInterpolationTypeValid() method" on page 131	When true, this property can be interpolated.
setInterpolationTypeAtKey()	"Property setInterpolationTypeAtKey() method" on page 140	Sets the interpolation type for a key.
keyInInterpolationType()	"Property keyInInterpolationType() method" on page 132	Gets the 'in' interpolation type for a key.
keyOutInterpolationType()	"Property keyOutInterpolationType() method" on page 133	Gets the 'out' interpolation type for a key.

Method	Reference	Description
`setSpatialTangentsAtKey()`	"Property setSpatialTangentsAtKey() method" on page 142	Sets the "in" and "out" tangent vectors for a key.
`keyInSpatialTangent()`	"Property keyInSpatialTangent() method" on page 132	Gets the "in" spatial tangent for a key.
`keyOutSpatialTangent()`	"Property keyOutSpatialTangent() method" on page 134	Gets the "out" spatial tangent for a key.
`setTemporalEaseAtKey()`	"Property setTemporalEaseAtKey() method" on page 144	Sets the "in" and "out" temporal ease for a key.
`keyInTemporalEase()`	"Property keyInTemporalEase() method" on page 133	Gets the "in" temporal ease for a key.
`keyOutTemporalEase()`	"Property keyOutTemporalEase() method" on page 134	Gets the "out" temporal ease for a key.
`setTemporalContinuousAtKey()`	"Property setTemporalContinuousAtKey() method" on page 143	Sets whether a keyframe has temporal continuity.
`keyTemporalContinuous()`	"Property keyTemporalContinuous() method" on page 136	Reports whether a keyframe has temporal continuity.
`setTemporalAutoBezierAtKey()`	"Property setTemporalAutoBezierAtKey() method" on page 143	Sets whether a keyframe has temporal auto-Bezier.
`keyTemporalAutoBezier()`	"Property keyTemporalAutoBezier() method" on page 136	Reports whether a keyframe has temporal auto-Bezier.
`setSpatialContinuousAtKey()`	"Property setSpatialContinuousAtKey() method" on page 142	Sets whether a keyframe has spatial continuity.
`keySpatialContinuous()`	"Property keySpatialContinuous() method" on page 135	Reports whether a keyframe has spatial continuity.
`setSpatialAutoBezierAtKey`	"Property setSpatialAutoBezierAtKey() method" on page 142	Sets whether a keyframe has spatial auto-Bezier.
`keySpatialAutoBezier()`	"Property keySpatialAutoBezier() method" on page 135	Reports whether a keyframe has spatial auto-Bezier.
`setRovingAtKey()`	"Property setRovingAtKey() method" on page 141	Sets whether a keyframe is roving.
`keyRoving()`	"Property keyRoving() method" on page 134	Reports whether a keyframe is roving.
`setSelectedAtKey()`	"Property setSelectedAtKey() method" on page 141	Sets whether a keyframe is selected.
`keySelected()`	"Property keySelected() method" on page 135	Reports whether a keyframe is selected.
`getSeparationFollower()`	"Property getSeparationFollower() method" on page 130	For a separated, multidimensional property, retrieves a specific follower property.

Example: Get and set the value of opacity

```
var myProperty = myLayer.opacity;
//opacity has propertyValueType of OneD, and is stored as a float
myProperty.setValue(50); // set opacity to 50%
// Variable myOpacity is a float value
var myOpacity = myProperty.value;
```

Example: Get and set the value of a position

```
var myProperty = myLayer.position;
//position has propertyValueType of ThreeD_SPATIAL, and is stored as an array of 3 floats
myProperty.setValue([10.0, 30.0, 0.0]);
// Variable myPosition is an array of 3 floats
var myPosition = myProperty.value;
```

Example: Change the value of a mask shape to be open instead of closed

```
var myMask = mylayer.mask(1);
var myProperty = myMask.maskPath;
myShape = myProperty.value;
myShape.closed = false;
myProperty.setValue(myShape);
```

Example: Get the value of a color at a particular time

A color is stored as an array of four floats, [r,g,b,opacity]. This sets the value of the red component of a light's color at time 4 to be half of that at time 2:

```
var myProperty = myLight.color;
var colorValue = myProperty.valueAtTime(2,true);
colorValue[0] = 0.5 * colorValue[0];
myProperty.setValueAtTime(4,colorValue);
```

Example: Check that a scale calculated by an expression at time 3.5 is the expected value of [10,50]

```
var myProperty = myLayer.scale;
// false value of preExpression means evaluate the expression
var scaleValue = myProperty.valueAtTime(3.5,false);
if (scaleValue[0] == 10 && scaleValue[1] == 50) {
  alert("hurray");
  }
else {
  alert("oops");
  }
```

Example: Keyframe a rotation from 0 to 90 and back again

The animation is 10 seconds, and the middle keyframe is at the 5 second mark. Rotation properties are stored as a OneD value.

```
myProperty = myLayer.rotation;
myProperty.setValueAtTime(0, 0);
myProperty.setValueAtTime(5, 90);
myProperty.setValueAtTime(10, 0);
```

Example: Change the keyframe values for the first three keyframes of some source text

```
myProperty = myTextLayer.sourceText;
if (myProperty.numKeys < 3) {
  alert("error, I thought there were 3 keyframes");
  }
else {
  myProperty.setValueAtKey(1, new TextDocument("key number 1"));
  myProperty.setValueAtKey(2, new TextDocument("key number 2"));
  myProperty.setValueAtKey(3, new TextDocument("key number 3"));
  }
```

Example: Set values using the convenience syntax for position, scale, color, or source text

```
// These two are equivalent. The second fills in a default of 0.
myLayer.position.setValue([20, 30, 0]);
myLayer.position.setValue([20, 30]);
// These two are equivalent. The second fills in a default of 100.
myLayer.scale.setValue([50, 50, 100]);
myLayer.scale.setValue([50, 50]);
// These two are equivalent. The second fills in a default of 1.0
myLight.color.setValue([.8, .3, .1, 1.0]);
myLight.color.setValue([.8, .3, .1]);
// These two are equivalent. The second creates a TextDocument
myTextLayer.sourceText.setValue(new TextDocument("foo"));
myTextLayer.sourceText.setValue("foo");
```

Property addKey() method

app.project.item(index).layer(index).*propertySpec*.addKey(*time*)

Description

Adds a new keyframe or marker to the named property at the specified time and returns the index of the new keyframe.

Parameters

time	The time, in seconds, at which to add the keyframe. A floating-point value. The beginning of the composition is 0.

Returns

Integer; the index of the new keyframe or marker.

Property canSetExpression attribute

app.project.item(index).layer(index).*propertySpec*.canSetExpression

Description

When true, the named property is of a type whose expression can be set by a script. See also “Property expression attribute” on page 129.

Type

Boolean; read-only.

Property canVaryOverTime attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.canVaryOverTime`

Description

When true, the named property can vary over time—that is, keyframe values or expressions can be written to this property.

Type

Boolean; read-only.

Property dimensionsSeparated attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.dimensionsSeparated`

Description

When true, the property's dimensions are represented as separate properties. For example, if the layer's position is represented as X Position and Y Position properties in the Timeline panel, the Position property has this attribute set to true.

NOTE: This attribute applies only when the `isSeparationLeader` *attribute is* `true`.

Type

Boolean; read/write.

Property expression attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.expression`

Description

The expression for the named property. Writeable only when `canSetExpression` for the named property is true. When you specify a value for this attribute, the string is evaluated.

- If the string contains a valid expression, `expressionEnabled` becomes true.
- If the string does not contain a valid expression, an error is generated, and `expressionEnabled` becomes false.
- If you set the attribute to the empty string, `expressionEnabled` becomes false, but no error is generated.

Type

String; read/write if `canSetExpression` for the named property is true.

Property expressionEnabled attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.expressionEnabled`

Description

When true, the named property uses its associated expression to generate a value. When false, the keyframe information or static value of the property is used. This attribute can be set to true only if `canSetExpression` for the named property is true and `expression` contains a valid expression string.

Type

Boolean; read/write.

Property expressionError attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.expressionError`

Description

Contains the error, if any, generated by evaluation of the string most recently set in `expression`. If no expression string has been specified, or if the last expression string evaluated without error, contains the empty string ("").

Type

String; read-only.

Property getSeparationFollower() method

`app.project.item(index).layer(index).`*`propertySpec`*`.getSeparationFollower(dim)`

Description

For a separated, multidimensional property, retrieves a specific follower property. For example, you can use this method on the Position property to access the separated X Position and Y Position properties.

NOTE: This attribute applies only when the `isSeparationLeader` *attribute is* `true`.

Parameters

`dim`	The dimension number (starting at 0).

Returns

Property object, or an error if the property is not multidimensional or does not have the specified dimension.

Property hasMax attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.hasMax`

Description

When true, there is a maximum permitted value for the named property; otherwise false.

Type

Boolean; read-only.

Property hasMin attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.hasMin`

Description

When true, there is a minimum permitted value for the named property; otherwise false.

Type

Boolean; read-only.

Property isInterpolationTypeValid() method

`app.project.item(index).layer(index).`*`propertySpec`*`.isInterpolationTypeValid(`*`type`*`)`

Description

Returns true if the named property can be interpolated using the specified keyframe interpolation type.

Parameters

`type`	A `KeyframeInterpolationType` enumerated value; one of:
	`KeyframeInterpolationType.LINEAR`
	`KeyframeInterpolationType.BEZIER`
	`KeyframeInterpolationType.HOLD`

Returns

Boolean.

Property isSeparationFollower attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.isSeparationFollower`

Description

When true, the property represents one of the separated dimensions for a multidimensional property. For example, the X Position property has this attribute set to true.

NOTE: The original, consolidated, multidimensional property is the "separation leader" and the new, separated, single-dimensional properties are its "separation followers".

Type

Boolean; read-only.

Property isSeparationLeader attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.isSeparationLeader`

Description

When true, the property is multidimensional and can be separated. For example, the Position property has this attribute set to true.

NOTE: The original, consolidated, multidimensional property is the "separation leader" and the new, separated, single-dimensional properties are its "separation followers".

Type

Boolean; read-only.

Property isSpatial attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.isSpatial`

Description

When true, the named property defines a spatial value. Examples are position and effect point controls.

Type

Boolean; read-only.

Property isTimeVarying attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.isTimeVarying`

Description

When true, the named property is time varying—that is, it has keyframes or an enabled expression. When this attribute is true, the attribute `canVaryOverTime` must also be true.

Type

Boolean; read-only.

Property keyInInterpolationType() method

`app.project.item(index).layer(index).`*`propertySpec`*`.keyInInterpolationType(`*`keyIndex`*`)`

Description

Returns the 'in' interpolation type for the specified keyframe.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearestKeyIndex` method.

Returns

A `KeyframeInterpolationType` enumerated value; one of:

`KeyframeInterpolationType.LINEAR`
`KeyframeInterpolationType.BEZIER`
`KeyframeInterpolationType.HOLD`

Property keyInSpatialTangent() method

`app.project.item(index).layer(index).`*`propertySpec`*`.keyInSpatialTangent(`*`keyIndex`*`)`

Description

Returns the incoming spatial tangent for the specified keyframe, if the named property is spatial (that is, the value type is `TwoD_SPATIAL` or `ThreeD_SPATIAL`).

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearestKeyIndex` method.

Returns

Array of floating-point values:

- If the property value type is `PropertyValueType.TwoD_SPATIAL`, the array contains 2 floating-point values.
- If the property value type is `PropertyValueType.ThreeD_SPATIAL`, the array contains 3 floating-point values.
- If the property value type is neither of these types, an exception is generated.

Property keyInTemporalEase() method

app.project.item(index).layer(index).*propertySpec*.keyInTemporalEase(*keyIndex*)

Description

Returns the incoming temporal ease for the specified keyframe.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearest-KeyIndex` method.

Returns

Array of KeyframeEase objects:

- If the property value type is `PropertyValueType.TwoD`, the array contains 2 objects.
- If the property value type is `PropertyValueType.ThreeD`, the array contains 3 objects.
- For any other value type, the array contains 1 object.

Property keyOutInterpolationType() method

app.project.item(index).layer(index).*propertySpec*.keyOutInterpolationType(*keyIndex*)

Description

Returns the outgoing interpolation type for the specified keyframe.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearest-KeyIndex` method.

Returns

A `KeyframeInterpolationType` enumerated value; one of:

KeyframeInterpolationType.LINEAR
KeyframeInterpolationType.BEZIER
KeyframeInterpolationType.HOLD

Property keyOutSpatialTangent() method

app.project.item(index).layer(index).*propertySpec*.keyOutSpatialTangent(*keyIndex*)

Description

Returns the outgoing spatial tangent for the specified keyframe.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearest-KeyIndex method.

Returns

Array of floating-point values:

- If the property value type is PropertyValueType.TwoD_SPATIAL, the array contains 2 floating-point values.
- If the property value type is PropertyValueType.ThreeD_SPATIAL, the array contains 3 floating-point values.
- If the property value type is neither of these types, an exception is generated.

Property keyOutTemporalEase() method

app.project.item*(index)*.layer*(index)*.*propertySpec*.keyOutTemporalEase(*keyIndex*)

Description

Returns the outgoing temporal ease for the specified keyframe.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearest-KeyIndex method.

Returns

Array of KeyframeEase objects:

- If the property value type is PropertyValueType.TwoD, the array contains 2 objects.
- If the property value type is PropertyValueType.ThreeD, the array contains 3 objects.
- For any other value type, the array contains 1 object.

Property keyRoving() method

app.project.item(index).layer(index).*propertySpec*.keyRoving(*keyIndex*)

Description

Returns true if the specified keyframe is roving. The first and last keyframe in a property cannot rove; if you try to set roving for one of these, the operation is ignored, and keyRoving() continues to return false.

If the property value type is neither TwoD_SPATIAL nor ThreeD_SPATIAL, an exception is generated.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.

Returns

Boolean.

Property keySelected() method

app.project.item(index).layer(index).*propertySpec*.keySelected(*keyIndex*)

Description

Returns true if the specified keyframe is selected.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.

Returns

Boolean.

Property keySpatialAutoBezier() method

app.project.item(index).layer(index).*propertySpec*.keySpatialAutoBezier(*keyIndex*)

Description

Returns true if the specified keyframe has spatial auto-Bezier interpolation. (This type of interpolation affects this keyframe only if keySpatialContinuous(*keyIndex*) is also true.)

If the property value type is neither TwoD_SPATIAL nor ThreeD_SPATIAL, an exception is generated.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.

Returns

Boolean.

Property keySpatialContinuous() method

app.project.item(index).layer(index).*propertySpec*.keySpatialContinuous(*keyIndex*)

Description

Returns true if the specified keyframe has spatial continuity.

If the property value type is neither TwoD_SPATIAL nor ThreeD_SPATIAL, an exception is generated.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.

Returns

Boolean.

Property keyTemporalAutoBezier() method

app.project.item(index).layer(index).*propertySpec*.keyTemporalAutoBezier(*keyIndex*)

Description

Returns true if the specified keyframe has temporal auto-Bezier interpolation.

Temporal auto-Bezier interpolation affects this keyframe only if the keyframe interpolation type is KeyframeInterpolationType.BEZIER for both keyInInterpolationType(*keyIndex*) and keyOutInterpolationType(*keyIndex*).

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.

Returns

Boolean.

Property keyTemporalContinuous() method

app.project.item(index).layer(index).*propertySpec*.keyTemporalContinuous(*keyIndex*)

Description

Returns true if the specified keyframe has temporal continuity.

Temporal continuity affects this keyframe only if keyframe interpolation type is KeyframeInterpolationType.BEZIER for both keyInInterpolationType(*keyIndex*) and keyOutInterpolationType(*keyIndex*).

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.

Returns

Boolean.

Property keyTime() method

app.project.item(index).layer(index).*propertySpec*.keyTime(*keyIndex*)
app.project.item(index).layer(index).*propertySpec*.keyTime(*markerComment*)

Description

Finds the specified keyframe or marker and returns the time at which it occurs.

If no keyframe or marker can be found that matches the argument, this method generates an exception, and an error is displayed.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearestKeyIndex` method.
`markerComment`	The comment string attached to a marker (see "MarkerValue comment attribute" on page 103).

Returns

Floating-point value.

Property keyValue() method

app.project.item(index).layer(index).*propertySpec*.keyValue(*keyIndex*)
app.project.item(index).layer(index).*propertySpec*.keyValue(*markerComment*)

Description

Finds the specified keyframe or marker and returns its current value.

If no keyframe or marker can be found that matches the argument, this method generates an exception, and an error is displayed.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearestKeyIndex` method.
`markerComment`	The comment string attached to a marker (see "MarkerValue comment attribute" on page 103).

Returns

Floating-point value for keyframes, MarkerValue object for markers.

Property maxValue attribute

app.project.item(index).layer(index).*propertySpec*.maxValue

Description

The maximum permitted value of the named property. If the `hasMax` attribute is false, an exception occurs, and an error is generated.

Type

Floating-point value; read-only.

Property minValue attribute

app.project.item(index).layer(index).*propertySpec*.minValue

Description

The minimum permitted value of the named property. If the `hasMin` attribute is false, an exception occurs, and an error is generated.

Type

Floating-point value; read-only.

Property nearestKeyIndex() method

`app.project.item(index).layer(index).`*`propertySpec`*`.nearestKeyIndex(`*`time`*`)`

Description

Returns the index of the keyframe nearest to the specified time.

Parameters

`time`	The time in seconds; a floating-point value. The beginning of the composition is 0.

Returns

Integer.

Property numKeys attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.numKeys`

Description

The number of keyframes in the named property. If the value is 0, the property is not being keyframed.

Type

Integer; read-only.

Property propertyIndex attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.propertyIndex`

Description

The position index of the named property. The first property is at index position 1.

Type

Integer; read-only.

Property propertyValueType attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.propertyValueType`

Description

The type of value stored in the named property. The `PropertyValueType` enumeration has one value for each type of data that can be stored in or retrieved from a property. Each type of data is stored and retrieved in a different kind of structure. All property objects store data according to one of these categories.

For example, a 3D spatial property (such as a layer's position) is stored as an array of three floating point values. When setting a value for position, pass in such an array, as follows:

```
mylayer.property("position").setValue([10,20,0]);
```

In contrast, a shape property (such as a layer's mask shape) is stored as a Shape object. When setting a value for a shape, pass a Shape object, as follows:

```
var myShape = new Shape();
myShape.vertices = [[0,0],[0,100],[100,100],[100,0]];
var myMask = mylayer.property("ADBE Mask Parade").property(1);
myMask.property("ADBE Mask Shape").setValue(myShape);
```

Type

A `PropertyValueType` enumerated value; read/write. One of:

`PropertyValueType.NO_VALUE`	Stores no data.
`PropertyValueType.ThreeD_SPATIAL`	Array of three floating-point positional values. For example, an Anchor Point value might be [10.0, 20.2, 0.0]
`PropertyValueType.ThreeD`	Array of three floating-point quantitative values. For example, a Scale value might be [100.0, 20.2, 0.0]
`PropertyValueType.TwoD_SPATIAL`	Array of 2 floating-point positional values For example, an Anchor Point value might be [5.1, 10.0]
`PropertyValueType.TwoD`	Array of 2 floating-point quantitative values. For example, a Scale value might be [5.1, 100.0]
`PropertyValueType.OneD`	A floating-point value.
`PropertyValueType.COLOR`	Array of 4 floating-point values in the range [0.0..1.0]. For example, [0.8, 0.3, 0.1, 1.0]
`PropertyValueType.CUSTOM_VALUE`	Custom property value, such as the Histogram property for the Levels effect.
`PropertyValueType.MARKER`	MarkerValue object; see "MarkerValue object" on page 102.
`PropertyValueType.LAYER_INDEX`	Integer; a value of 0 means no layer.
`PropertyValueType.MASK_INDEX`	Integer; a value of 0 means no mask.
`PropertyValueType.SHAPE`	Shape object; see "Shape object" on page 172.
`PropertyValueType.TEXT_DOCUMENT`	TextDocument object; see "TextDocument object" on page 182.

Property removeKey() method

`app.project.item(index).layer(index).`*`propertySpec`*`.removeKey(`*`keyIndex`*`)`

Description

Removes the specified keyframe from the named property. If no keyframe with the specified index exists, generates an exception and displays an error.

When a keyframe is removed, the remaining index numbers change. To remove more than one keyframe, you must start with the highest index number and work down to the lowest to ensure that the remaining indices reference the same keyframe after each removal.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearest-KeyIndex` method.

Returns

Nothing.

Property selectedKeys attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.selectedKeys`

Description

The indices of all the selected keyframes in the named property. If no keyframes are selected, or if the property has no keyframes, returns an empty array.

Type

Array of integers; read-only.

Property separationDimension attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.separationDimension`

Description

For a separated follower, the dimension number it represents in the multidimensional leader. The first dimension starts at 0. For example, the Y Position property has a `separationDimension` value of 1; X Position has a value of 0.

Type

Integer; read-only.

Property separationLeader attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.separationLeader`

Description

The original multidimensional property for this separated follower. For example, if the current property is Y Position, this attribute's value points to the Position property.

NOTE: The original, consolidated, multidimensional property is the "separation leader" and the new, separated, single-dimensional properties are its "separation followers".

Type

Property object; read-only.

Property setInterpolationTypeAtKey() method

`app.project.item(index).layer(index).`*`propertySpec`*`.setInterpolationTypeAtKey(`*`keyIndex`*`, `*`inType, outType`*`)`

Description

Sets the 'in' and 'out' interpolation types for the specified keyframe.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.
inType	The incoming interpolation type. A KeyframeInterpolationType enumerated value; one of: KeyframeInterpolationType.LINEAR KeyframeInterpolationType.BEZIER KeyframeInterpolationType.HOLD
outType	(Optional) The outgoing interpolation type. If not supplied, the 'out' type is set to the inType value. A KeyframeInterpolationType enumerated value; one of: KeyframeInterpolationType.LINEAR KeyframeInterpolationType.BEZIER KeyframeInterpolationType.HOLD

Returns

Nothing.

Property setRovingAtKey() method

app.project.item(index).layer(index).*propertySpec*.setRovingAtKey(*keyIndex, newVal*)

Description

Turns roving on or off for the specified keyframe. The first and last keyframe in a property cannot rove; if you try to set roving for one of these, the operation is ignored, and keyRoving() continues to return false.

If the property value type is neither TwoD_SPATIAL nor ThreeD_SPATIAL, an exception is generated.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKey-Index method.
newVal	True to turn roving on, false to turn roving off.

Returns

Nothing.

Property setSelectedAtKey() method

app.project.item(index).layer(index).*propertySpec*.setSelectedAtKey(*keyIndex, onOff*)

Description

Selects or deselects the specified keyframe.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearest-KeyIndex method.
onOff	True to select the keyframe, false to deselect it.

Returns

Nothing.

Property setSpatialAutoBezierAtKey() method

`app.project.item(index).layer(index).`*`propertySpec`*`.setSpatialAutoBezierAtKey(keyIndex, newVal)`

Description

Turns spatial auto-Bezier interpolation on or off for the specified keyframe.

If the property value type is neither `TwoD_SPATIAL` nor `ThreeD_SPATIAL`, an exception is generated.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearest-KeyIndex` method.
newVal	True to turn spatial auto-Bezier on, false to turn it off.

Returns

Nothing.

Property setSpatialContinuousAtKey() method

`app.project.item(index).layer(index).`*`propertySpec`*`.setSpatialContinuousAtKey(`*`keyIndex, newVal`*`)`

Description

Turns spatial continuity on or off for the specified keyframe.

If the property value type is neither `TwoD_SPATIAL` nor `ThreeD_SPATIAL`, an exception is generated.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearest-KeyIndex` method.
newVal	True to turn spatial continuity on, false to turn it off.

Returns

Nothing.

Property setSpatialTangentsAtKey() method

`app.project.item(index).layer(index).`*`propertySpec`*`.setSpatialTangentsAtKey(`*`keyIndex, inTangent, outTangent`*`)`

Description

Sets the incoming and outgoing tangent vectors for the specified keyframe.

If the property value type is neither `TwoD_SPATIAL` nor `ThreeD_SPATIAL`, an exception is generated.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearest-KeyIndex method.
inTangent	The incoming tangent vector. An array of 2 or 3 floating-point values. • If the property value type is PropertyValueType.TwoD_SPATIAL, the array contains 2 values. • If the property value type is PropertyValueType.ThreeD_SPATIAL, the array contains 3 values.
outTangent	(Optional) The outgoing tangent vector. If not supplied, the 'out' tangent is set to the inTangent value. An array of 2 or 3 floating-point values. • If the property value type is PropertyValueType.TwoD_SPATIAL, the array contains 2 values. • If the property value type is PropertyValueType.ThreeD_SPATIAL, the array contains 3 values.

Returns

Nothing.

Property setTemporalAutoBezierAtKey() method

app.project.item(index).layer(index).*propertySpec*.setTemporalAutoBezierAtKey(*keyIndex, newVal*)

Description

Turns temporal auto-Bezier interpolation on or off for the specified keyframe. When this is turned on, it affects this keyframe only if keySpatialContinuous(*keyIndex*) is also true.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearest-KeyIndex method.
newVal	True to turn temporal auto-Bezier on, false to turn it off.

Returns

Nothing.

Property setTemporalContinuousAtKey() method

app.project.item(index).layer(index).*propertySpec*.setTemporalContinuousAtKey(*keyIndex, newVal*)

Description

Turns temporal continuity on or off for the specified keyframe.

When temporal continuity is turned on, it affects this keyframe only if the keyframe interpolation type is KeyframeInterpolationType.BEZIER for both keyInInterpolationType(*keyIndex*) and keyOutInterpolationType(*keyIndex*).

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearest-KeyIndex method.
newVal	True to turn temporal continuity on, false to turn it off.

Returns

Nothing.

Property setTemporalEaseAtKey() method

app.project.item(index).layer(index).*propertySpec*.setTemporalEaseAtKey(*keyIndex*, *inTemporalEase*, *outTemporalEase*)

Description

Sets the incoming and outgoing temporal ease for the specified keyframe. See “KeyframeEase object” on page 84.

Parameters

keyIndex	The index for the keyframe. An integer in the range [1..numKeys], as returned by the addKey or nearestKeyIndex method.
inTemporalEase	The incoming temporal ease. An array of 1, 2, or 3 KeyframeEase objects. • If the property value type is PropertyValueType.TwoD, the array contains 2 objects. • If the property value type is PropertyValueType.ThreeD, the array contains 3 objects. • For all other value types, the array contains 1 object.
outTemporalEase	(Optional) The outgoing temporal ease. If not supplied, the outgoing ease is set to the inTemporalEase value. An array of 1, 2, or 3 KeyframeEase objects. • If the property value type is PropertyValueType.TwoD, the array contains 2 objects. • If the property value type is PropertyValueType.ThreeD, the array contains 3 objects. • For all other value types, the array contains 1 object.

Returns

Nothing.

Property setValue() method

app.project.item(index).layer(index).*propertySpec*.setValue(*newValue*)

Description

Sets the static value of a property that has no keyframes.

If the named property has keyframes, this method generates an exception and displays an error. To set the value of a property with keyframes, use “Property setValueAtTime() method” on page 145 or “Property setValueAtKey() method” on page 145.

Parameters

newValue	A value appropriate for the type of property being set; see “Property propertyValueType attribute” on page 138.

Returns

Nothing.

Property setValueAtKey() method

`app.project.item(index).layer(index).`*`propertySpec`*`.setValueAtKey(`*`keyIndex, newValue`*`)`

Description

Finds the specified keyframe and sets its value.

If the named property has no keyframes, or no keyframe with the specified index, this method generates an exception and displays an error.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range [1..numKeys], as returned by the `addKey` or `nearest-KeyIndex` method.
`newValue`	A value appropriate for the type of property being set; see "Property propertyValueType attribute" on page 138.

Returns

Nothing.

Property setValueAtTime() method

`app.project.item(index).layer(index).`*`propertySpec`*`.setValueAtTime(`*`time, newValue`*`)`

Description

Sets the value of a keyframe at the specified time. Creates a new keyframe for the named property, if one does not currently exist for the specified time, and sets its value.

Parameters

`time`	The time in seconds, a floating-point value. The beginning of the composition is 0.
`newValue`	A value appropriate for the type of property being set; see "Property propertyValueType attri-bute" on page 138.

Returns

Nothing.

Property setValuesAtTimes() method

`app.project.item(index).layer(index).`*`propertySpec`*`.setValuesAtTimes(`*`times, newValues`*`)`

Description

Sets values for a set of keyframes at specified of times. Creates a new keyframe for the named property, if one does not currently exist for a specified time, and sets its value.

Times and values are expressed as arrays; the arrays must be of the same length.

Parameters

`times`	An array of times, in seconds. Each time is a floating-point value. The beginning of the compo-sition is 0.

`newValues`	A array of values appropriate for the type of property being set; see “Property propertyValueType attribute” on page 138.

Returns

Nothing.

Property unitsText attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.unitsText`

Description

The text description of the units in which the value is expressed.

Type

String; read-only.

Property value attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.value`

Description

The value of the named property at the current time.

- If `expressionEnabled` is true, returns the evaluated expression value.
- If there are keyframes, returns the keyframed value at the current time.
- Otherwise, returns the static value.

The type of value returned depends on the property value type. See examples for “Property object” on page 124.

Type

A value appropriate for the type of the property (see “Property propertyValueType attribute” on page 138); read-only.

Property valueAtTime() method

`app.project.item(index).layer(index).`*`propertySpec`*`.valueAtTime(`*`time, preExpression`*`)`

Description

The value of the named property as evaluated at the specified time.

Note that the type of value returned is not made explicit; it will be of a different type, depending on the property evaluated.

Parameters

`time`	The time in seconds; a floating-point value. The beginning of the composition is 0.
`preExpression`	If the property has an expression and this is true, return the value for the specified time without applying the expression to it. When false, return the result of evaluating the expression for the specified time. Ignored if the property does not have an associated expression.

Returns

A value appropriate for the type of the property (see “Property propertyValueType attribute” on page 138).

PropertyBase object

app.project.item(*index*).layer(*index*).*propertySpec*

Description

Properties are accessed by name through layers, using various kinds of expression syntax, as controlled by application preferences. For example, the following are all ways of access properties in the Effects group:

```
var effect1 = app.project.item(1).layer(1).effect("Add Grain")("Viewing Mode");
var effect1again = app.project.item(1).layer(1).effect.addGrain.viewingMode;
var effect1againtoo = app.project.item(1).layer(1)("Effects").addGrain.viewingMode;
var effect1againtoo2 = app.project.item(1).layer(1)("Effects")("Add Grain")("Viewing Mode");
```

See also "PropertyGroup property() method" on page 157.

- PropertyBase is the base class for both Property and PropertyGroup, so PropertyBase attributes and methods are available when working with properties and property groups. See "Property object" on page 124 and "PropertyGroup object" on page 155.

Reference invalidation

When something occurs that changes an object sufficiently for the reference to become invalid, script references to that object can generate errors. In simple cases this is straightforward. For example, if you delete an object, a reference to the deleted object generates the warning "Object is Invalid":

```
var layer1 = app.project.item(1).layer(1);
layer1.remove();
alert(layer1.name); // invalid reference to deleted object
```

Similarly, if you reference an AE property in a deleted object, the warning occurs:

```
var layer1 = app.project.item(1).layer(1);
var layer1position = layer1.transform.position;
layer1.remove();
alert(layer1position.value); // invalid reference to property in selected object
```

A less straightforward case is when a property is removed from a property group. In this case, After Effects generates the "Object is Invalid" error when you subsequently reference that item or other items in the group, because their index positions have changed. For example:

```
var effect1 = app.project.item(1).layer(1).effect(1);
var effect2 = app.project.item(1).layer(1).effect(2);
var effect2param = app.project.item(1).layer(1).effect(2).blendWithOriginal;
effect1.remove();
alert(effect2.name); // invalid reference because group index positions have changed
```

Attributes

Attribute	Reference	Description
`name`	"PropertyBase name attribute" on page 152	Name of the property.
`matchName`	"PropertyBase matchName attribute" on page 152	A special name for the property used to build unique naming paths.

Attribute	Reference	Description
propertyIndex	"PropertyBase propertyIndex attribute" on page 153	Index of this property within its parent group.
propertyDepth	"PropertyBase propertyDepth attribute" on page 153	The number of levels of parent groups between this property and the containing layer.
propertyType	"PropertyBase propertyType attribute" on page 154	The property type.
parentProperty	"PropertyBase parentProperty attribute" on page 153	The immediate parent group of this property.
isModified	"PropertyBase isModified attribute" on page 151	When true, the property has been changed since its creation.
canSetEnabled	"PropertyBase canSetEnabled attribute" on page 150	When true, the user interface displays an eyeball icon for this property.
enabled	"PropertyBase enabled attribute" on page 151	When true, this property is enabled.
active	"PropertyBase active attribute" on page 149	When true, this property is active.
elided	"PropertyBase elided attribute" on page 150	When true, this property is not displayed in the user interface.
isEffect	"PropertyBase isEffect attribute" on page 151	When true, this property is an effect.
isMask	"PropertyBase isMask attribute" on page 151	When true, this property is a mask.
selected	"PropertyBase selected attribute" on page 154	When true, this property is selected.

Methods

Method	Reference	Description
propertyGroup()	"PropertyBase propertyGroup() method" on page 153	Gets the parent group for this property.
remove()	"PropertyBase remove() method" on page 154	Removes this from the project.
moveTo()	"PropertyBase moveTo() method" on page 152	Moves this property to a new position in its parent group.
duplicate()	"PropertyBase duplicate() method" on page 150	Duplicates this property object.

PropertyBase active attribute

app.project.item(index).layer(index).*propertySpec*.active

Description

When true, this property is active. For a layer, this corresponds to the setting of the eyeball icon and if the current time is between the layer's in and out points. For an effect and all properties, it is the same as the enabled attribute, except that it's read-only.

Type

Boolean; read-only.

PropertyBase canSetEnabled attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.canSetEnabled`

Description

When true, you can set the `enabled` attribute value. Generally, this is true if the user interface displays an eyeball icon for this property; it is true for all layers.

Type

Boolean; read-only.

PropertyBase duplicate() method

`app.project.item(index).layer(index).`*`propertySpec`*`.duplicate()`

Description

If this property is a child of an indexed group, creates and returns a new PropertyBase object with the same attribute values as this one.

If this property is not a child of an indexed group, the method generates an exception and displays an error.

An indexed group has the type `PropertyType.INDEXED_GROUP`; see "PropertyBase propertyType attribute" on page 154.

Parameters

None.

Returns

PropertyBase object.

PropertyBase elided attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.elided`

Description

When true, this property is a group used to organize other properties. The property is not displayed in the user interface and its child properties are not indented in the Timeline panel.

For example, for a text layer with two animators and no properties twirled down, you might see:

```
Text
Path Options
More Options
Animator 1
Animator 2
```

In this example, "Animator 1" and "Animator 2" are contained in a PropertyBase called "Text Animators." This parent group is not displayed in the user interface, and so the two child properties are not indented in the Timeline panel.

Type

Boolean; read-only.

PropertyBase enabled attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.enabled`

Description

When true, this property is enabled. It corresponds to the setting of the eyeball icon, if there is one; otherwise, the default is true.

Type

Boolean; read/write if `canSetEnabled` is true, read-only if `canSetEnabled` is false.

PropertyBase isEffect attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.isEffect`

Description

When true, this property is an effect PropertyGroup.

Type

Boolean; read-only.

PropertyBase isMask attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.isMask`

Description

When true, this property is a mask PropertyGroup.

Type

Boolean; read-only.

PropertyBase isModified attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.isModified`

Description

When true, this property has been changed since its creation.

Type

Boolean; read-only.

PropertyBase matchName attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.matchName`

Description

A special name for the property used to build unique naming paths. The match name is not displayed, but you can refer to it in scripts. Every property has a unique match-name identifier. Match names are stable from version to version regardless of the display name (the `name` attribute value) or any changes to the application. Unlike the display name, it is not localized.

An indexed group may not have a `name` value, but always has a `matchName` value. (An indexed group has the type `PropertyType.INDEXED_GROUP`; see "PropertyBase propertyType attribute" on page 154.)

Type

String; read-only.

PropertyBase moveTo() method

`app.project.item(index).layer(index).`*`propertySpec`*`.moveTo(`*`newIndex`*`)`

Description

Moves this property to a new position in its parent property group.

This method is valid only for children of indexed groups; if it is not, or if the index value is not valid, the method generates an exception and displays an error. (An indexed group has the type `Proper-tyType.INDEXED_GROUP`; see "PropertyBase propertyType attribute" on page 154.)

NOTE: Using this method invalidates existing references to other children in the same indexed group. For example, if you have three effects on a layer, each effect assigned to a different variable, moving one of the effects invalidates the references for all of these variables. You will need to reassign them.

Parameters

`newIndex`	The new index position at which to place this property in its group. An integer.

Returns

Nothing.

PropertyBase name attribute

`app.project.item(index).layer(index).`*`propertySpec`*`.name`

Description

The display name of the property. (Compare "PropertyBase matchName attribute" on page 152.)

It is an error to set the `name` value if the property is not a child of an indexed group (that is, a property group that has the type `PropertyType.INDEXED_GROUP`; see "PropertyBase propertyType attribute" on page 154).

Type

String; read/write for a child of an indexed group; otherwise read-only.

PropertyBase parentProperty attribute

app.project.item(index).layer(index).*propertySpec*.parentProperty

Description

The property group that is the immediate parent of this property, or null if this PropertyBase is a layer.

Type

PropertyGroup object or null; read-only.

PropertyBase propertyDepth attribute

app.project.item(index).layer(index).*propertySpec*.propertyDepth

Description

The number of levels of parent groups between this property and the containing layer. The value 0 for a layer.

Type

Integer; read-only.

PropertyBase propertyGroup() method

app.project.item(index).layer(index).*propertySpec*.propertyGroup()

app.project.item(index).layer(index).*propertySpec*.propertyGroup(*countUp*)

Description

Gets the PropertyGroup object for an ancestor group of this property at a specified level of the parent-child hierarchy.

Parameters

countUp	Optional. The number of levels to ascend within the parent-child hierarchy. An integer in the range [1..propertyDepth]. Default is 1, which gets the immediate parent.

Returns

PropertyGroup object, or null if the count reaches the containing layer.

PropertyBase propertyIndex attribute

app.project.item(index).layer(index).*propertySpec*.propertyIndex

Description

The position index of this property within its parent group, if it is a child of an indexed group (a property group that has the type PropertyType.INDEXED_GROUP; see “PropertyBase propertyType attribute” on page 154).

Type

Integer; read-only.

PropertyBase propertyType attribute

app.project.item(index).layer(index).*propertySpec*.propertyType

Description

The type of this property.

Type

A PropertyType enumerated value; read/write. One of:

PropertyType.PROPERTY	A single property such as position or zoom.
PropertyType.INDEXED_GROUP	A property group whose members have an editable name and an index. Effects and masks are indexed groups. For example, the masks property of a layer refers to a variable number of individual masks by index number.
PropertyType.NAMED_GROUP	A property group in which the member names are not editable. Layers are named groups.

PropertyBase remove() method

app.project.item(index).layer(index).*propertySpec*.remove()

Description

Removes this property from its parent group. If this is a property group, it removes the child properties as well.

This method is valid only for children of indexed groups; if it is not, or if the index value is not valid, the method generates an exception and displays an error. (An indexed group has the type PropertyType.INDEXED_GROUP; see "PropertyBase propertyType attribute" on page 154.)

This method can be called on a text animation property (that is, any animator that has been set to a text layer).

Parameters

None.

Returns

Nothing.

PropertyBase selected attribute

app.project.item(index).layer(index).*propertySpec*.selected

Description

When true, this property is selected. Set to true to select the property, or to false to deselect it.

Sampling this attribute repeatedly for a large number of properties can slow down system performance. To read the full set of selected properties of a composition or layer, use the selectedProperties attribute of a Comp or Layer object.

Type

Boolean; read/write.

PropertyGroup object

`app.project.item(index).layer(index).`*`propertyGroupSpec`*

Description

The PropertyGroup object represents a group of properties. It can contain Property objects and other PropertyGroup objects. Property groups can be nested to provide a parent-child hierarchy, with a Layer object at the top (root) down to a single Property object, such as the mask feather of the third mask. To traverse the group hierarchy, use PropertyBase methods and attributes; see “PropertyBase propertyGroup() method” on page 153.

For examples of how to access properties and property groups, see “PropertyBase object” on page 148.

- PropertyGroup is a subclass of PropertyBase. All methods and attributes of PropertyBase, in addition to those listed below, are available when working with PropertyGroup. See “PropertyBase object” on page 148.
- PropertyGroup is a base class for MaskPropertyGroup. PropertyGroup attributes and methods are available when working with mask groups. See “MaskPropertyGroup object” on page 106.

Attributes

Attribute	Reference	Description
`numProperties`	“PropertyGroup numProperties attribute” on page 156	The number of indexed properties in the group.

Methods

Method	Reference	Description
`property()`	“PropertyGroup property() method” on page 157	Gets a member property or group.
`canAddProperty()`	“PropertyGroup canAddProperty() method” on page 156	Reports whether a property can be added to the group.
`addProperty()`	“PropertyGroup addProperty() method” on page 155	Adds a property to the group.

PropertyGroup addProperty() method

`app.project.item(index).layer(index).`*`propertyGroupSpec`*`.addProperty(`*`name`*`)`

Description

Creates and returns a PropertyBase object with the specified name, and adds it to this group.

In general, you can only add properties to an indexed group (a property group that has the type `PropertyType.INDEXED_GROUP`; see “PropertyBase propertyType attribute” on page 154) The only exception is a text animator property, which can be added to a named group (a property group that has the type `PropertyType.NAMED_GROUP`).

If this method cannot create a property with the specified name, it generates an exception. To check that you can add a particular property to this group, call `canAddProperty` before calling this method. (See “Property-Group canAddProperty() method” on page 156.)

Parameters

`name`	The display name or match name of the property to add. (See “PropertyBase matchName attribute” on page 152). The following names are supported: • Any match name for a property that can be added through the user interface. For example, “ADBE Mask Atom”, “ADBE Paint Atom”, “ADBE Text Position”, “ADBE Text Anchor Point”. • When adding to an ADBE Mask Parade: “ADBE Mask Atom”, “Mask”. • When adding to an ADBE Effect Parade, any effect by match name, such as “ADBE Bulge”, “ADBE Glo2”, “APC Vegas”. • Any effect by display name, such as “Bulge”, “Glow”, “Vegas”. • For text animators, “ADBE Text Animator”. • For selectors, Range Selector has the name “ADBE Text Selector”, Wiggly Selector has the name “ADBE Text Wiggly Selector”, and Expression Selector has the name “ADBE Text Expressible Selector”.

Returns

PropertyBase object.

PropertyGroup canAddProperty() method

`app.project.item(index).layer(index).`*`propertyGroupSpec`*`.canAddProperty(`*`name`*`)`

Description

Returns true if a property with the given name can be added to this property group. For example, you can only add mask to a mask group. The only legal input arguments are “mask” or “ADBE Mask Atom”.

```
maskGroup.canAddProperty("mask"); //returns true
maskGroup.canAddProperty("ADBE Mask Atom"); //returns true
maskGroup.canAddProperty("blend"); // returns false
```

Parameters

`name`	The display name or match name of the property to be checked. (See “PropertyGroup addProperty() method” on page 155).

Returns

Boolean.

PropertyGroup numProperties attribute

`app.project.item(index).layer(index).`*`propertyGroupSpec`*`.numProperties`

Description

The number of indexed properties in this group.

For layers, this method returns a value of 3, corresponding to the mask, effect, and motion tracker groups, which are the indexed groups within the layer. However, layers also have many other properties available only by name; see the “PropertyGroup property() method” on page 157.

Type

Integer; read-only.

PropertyGroup property() method

app.project.item(index).layer(index).*propertyGroupSpec*.property*(index)*
app.project.item(index).layer(index).*propertyGroupSpec*.property*(name)*

Description

Finds and returns a child property of this group, as specified by either its index or name.

A name specification can use the same syntax that is available with expressions. The following are all allowed and are equivalent:

```
mylayer.position
mylayer("position")
mylayer.property("position")
mylayer(1)
mylayer.property(1)
```

Some properties of a layer, such as position and zoom, can be accessed only by name.

When using the name to find a property that is multiple levels down, you must make more than one call to this method. For example, the following call searches two levels down, and returns the first mask in the mask group:

```
myLayer.property("ADBE Masks").property(1)
```

Parameters

index	The index for the child property, in this is an indexed group. An integer in the range [0..numProperties].
name	The name of the child property. This can be: • Any match name • Any name in expression “parenthesis style” syntax, meaning the display name or the compact English name • Any name in expression “intercap style” syntax For supported property names, see the table below.

Returns

PropertyBase object or null if no child property with the specified string name is found.

Properties accessible by name

From any Layer	• "ADBE Mask Parade", or “Masks” • "ADBE Effect Parade", or “Effects” • "ADBE MTrackers", or “Motion Trackers”

From an AVLayer	• "Anchor Point" or "anchorPoint" • "Position" or "position" • "Scale" or "scale" • "Rotation" or "rotation" • "Z Rotation" or "zRotation" or "Rotation Z" or "rotationZ" • "Opacity" or "opacity" • "Marker" or "marker"
From an AVLayer with a non-still source	• "Time Remap" or "timeRemapEnabled"
From an AVLayer with an audio component	• "Audio Levels" or "audioLevels"
From a camera layer	• "Zoom" or "zoom" • "Depth of Field" or "depthOfField" • "Focus Distance" or "focusDistance" • "Aperture" or "aperture" • "Blur Level" or "blurLevel"
From a light layer	• "Intensity" or "intensity" • "Color" or "color" • "Cone Angle" or "coneAngle" • "Cone Feather" or "coneFeather" • "Shadow Darkness" or "shadowDarkness" • "Shadow Diffusion" or "shadowDiffusion" • "Casts Shadows" or "castsShadows"
From a 3D layer	• "Accepts Shadows" or "acceptsShadows" • "Accepts Lights" or "acceptsLights" • "Ambient" or "ambient" • "Diffuse" or "diffuse" • "Specular" or "specular" (these are for the Specular Intensity property) • "Shininess" or "shininess" (these are for the Specular Shininess property) • "Casts Shadows" or "castsShadows" • "Light Transmission" or "lightTransmission" • "Metal" or "metal"
From a camera, light or 3D layer	• "X Rotation" or "xRotation" or "Rotation X" or "rotationX" • "Y Rotation" or "yRotation" or "Rotation Y" or "rotationY" • "Orientation" or "orientation"
From a text layer	• "Source Text" or "sourceText" or "Text" or "text"
From a PropertyGroup "ADBE Mask Parade"	• "ADBE Mask Atom"
From a PropertyGroup "ADBE Mask Atom"	• "ADBE Mask Shape", or “maskShape”, or “maskPath” • "ADBE Mask Feather", or “maskFeather” • "ADBE Mask Opacity", or “maskOpacity” • "ADBE Mask Offset", or “maskOffset”

Examples

1 If a layer named “myLayer” has a Box Blur effect, you can retrieve the effect in any of the following ways:

```
myLayer.property(“Effects”).property(“Box Blur”);
myLayer.property(“Effects”).property(“boxBlur”);
myLayer.property(“Effects”).property(“ADBE Box Blur”);
```

2 If a layer named “myLayer” has a mask named “Mask 1” you can retrieve it as follows:

```
myLayer.property(“Masks”).property(“Mask 1”);
```

3 To get a Bulge Center value from a Bulge effect, you can use either of the following:

```
myLayer.property(“Effects”).property(“Bulge”).property(“Bulge Center”);
myLayer.property(“Effects”).property(“Bulge”).property(“bulgeCenter”);
```

RenderQueue object

`app.project.renderQueue`

Description

The RenderQueue object represents the render automation process, the data and functionality that is available through the Render Queue panel of a particular After Effects project. Attributes provide access to items in the render queue and their render status. Methods can start, pause, and stop the rendering process.

The RenderQueueItem object provides access to the specific settings for an item to be rendered. See "Render-QueueItem object" on page 163.

Attributes

Attribute	Reference	Description
`rendering`	"RenderQueue rendering attribute" on page 162	When true, a render is in progress.
`numItems`	"RenderQueue numItems attribute" on page 161	The total number of items in the render queue.
`items`	"RenderQueue items attribute" on page 161	The collection of items in the render queue.

Methods

Method	Reference	Description
`showWindow()`	"RenderQueue showWindow() method" on page 162	Show or hides the Render Queue panel.
`render()`	"RenderQueue render() method" on page 161	Starts the rendering process; does not return until render is complete.
`pauseRendering()`	"RenderQueue pauseRendering() method" on page 161	Pauses or restarts the rendering process.
`stopRendering()`	"RenderQueue stopRendering() method" on page 162	Stops the rendering process.
`item()`	"RenderQueue item() method" on page 160	Gets a render-queue item from the collection.

RenderQueue item() method

`app.project.renderQueue.item(`*`index`*`)`

Description

Gets a specified item from the `items` collection.

Parameters

`index`	The position index of the item. An integer in the range [0..`numItems`].

Returns

RenderQueueItem object.

RenderQueue items attribute

`app.project.renderQueue.items`

Description

A collection of all items in the render queue. See “RenderQueueItem object” on page 163.

Type

RQItemCollection object; read-only.

RenderQueue numItems attribute

`app.project.renderQueue.numItems`

Description

The total number of items in the render queue.

Type

Integer; read-only.

RenderQueue pauseRendering() method

`app.project.renderQueue.pauseRendering(`*`pause`*`)`

Description

Pauses the current rendering process, or continues a paused rendering process. This is the same as clicking Pause in the Render Queue panel during a render. You can call this method from an `onStatusChanged` or `onError` callback. See “RenderQueueItem onStatusChanged attribute” on page 165 and “Application onError attribute” on page 24.

Parameters

`pause`	True to pause a current render process, false to continue a paused render.

Returns

Nothing.

RenderQueue render() method

`app.project.renderQueue.render()`

Description

Starts the rendering process. This is the same as clicking Render in the Render Queue panel. The method does not return until the render process is complete. To pause or stop the rendering process, call `pauseRendering()` or `stopRendering()` from an `onError` or `onStatusChanged` callback.

- To respond to errors during the rendering process, define a callback function in `app.onError`; see “Application onError attribute” on page 24.
- To respond to changes in the status of a particular item while the render is progressing, define a callback function in `RenderQueueItem.onStatusChanged` in the associated RenderQueueItem object; see “RenderQueueItem onStatusChanged attribute” on page 165.

Parameters

None.

Returns

Nothing.

RenderQueue rendering attribute

`app.project.renderQueue.rendering`

Description

When true, the rendering process is in progress or paused. When false, it is stopped.

Type

Boolean; read-only.

RenderQueue showWindow() method

`app.project.renderQueue.showWindow(doShow)`

Description

Shows or hides the Render Queue panel.

Parameters

`doShow`	When true, show the Render Queue panel. When false, hide it.

Returns

Nothing.

RenderQueue stopRendering() method

`app.project.renderQueue.stopRendering()`

Description

Stops the rendering process. This is the same as clicking Stop in the Render Queue panel during a render. You can call this method from an `onStatusChanged` or `onError` callback. See “RenderQueueItem onStatus-Changed attribute” on page 165 and “Application onError attribute” on page 24.

Parameters

None.

Returns

Nothing.

RenderQueueItem object

`app.project.renderQueue.item(`*`index`*`)`

Description

The RenderQueueItem object represents an individual item in the render queue. It provides access to the specific settings for an item to be rendered. Create the object by adding a composition to the Render Queue with the RQItemCollection object; see "RQItemCollection add() method" on page 169.

Attributes

Attribute	Reference	Description
numOutputModules	"RenderQueueItem numOutputModules attribute" on page 165	The total number of Output Modules assigned to the item.
render	"RenderQueueItem render attribute" on page 166	When true, this item is rendered when the queue is started.
startTime	"RenderQueueItem startTime attribute" on page 167	The time when rendering began for the item.
elapsedSeconds	"RenderQueueItem elapsedSeconds attribute" on page 164	The time elapsed in the current rendering of this item.
timeSpanStart	"RenderQueueItem timeSpanStart attribute" on page 168	The start time in the composition to be rendered.
timeSpanDuration	"RenderQueueItem timeSpanDuration attribute" on page 168	The duration of the composition to be rendered.
skipFrames	"RenderQueueItem skipFrames attribute" on page 167	The number of frames to skip when rendering this item.
comp	"RenderQueueItem comp attribute" on page 164	The composition to be rendered by this item.
outputModules	"RenderQueueItem outputModules attribute" on page 166	The collection of Output Modules for this item.
templates	"RenderQueueItem templates attribute" on page 168	A set of Render Settings templates.
status	"RenderQueueItem status attribute" on page 167	The current rendering status of the item.
onStatusChanged	"RenderQueueItem onStatusChanged attribute" on page 165	A callback function that is called during the rendering process when the status of the item changes.
logType	"RenderQueueItem logType attribute" on page 165	A log type for this item.

Methods

Method	Reference	Description
outputModule()	"RenderQueueItem outputModule() method" on page 166	Gets an Output Module for the item.
remove()	"RenderQueueItem remove() method" on page 166	Removes the item from the render queue.
saveAsTemplate()	"RenderQueueItem saveAsTemplate() method" on page 167	Saves a new Render Settings template.

Method	Reference	Description
`applyTemplate()`	"RenderQueueItem applyTemplate() method" on page 164	Applies a Render Settings template.
`duplicate`	"RenderQueueItem duplicate() method" on page 164	Duplicates this item.

RenderQueueItem applyTemplate() method

`app.project.renderQueue.item(index).applyTemplate(templateName)`

Description

Applies a Render Settings template to the item. See also "RenderQueueItem saveAsTemplate() method" on page 167 and "RenderQueueItem templates attribute" on page 168.

Parameters

`templateName`	A string containing the name of the template to apply.

Returns

Nothing.

RenderQueueItem comp attribute

`app.project.renderQueue.item(index).comp`

Description

The composition that will be rendered by this render-queue item. To change the composition, you must delete this render-queue item and create a new one.

Type

CompItem object; read-only.

RenderQueueItem duplicate() method

`app.project.renderQueue.item(index).duplicate()`

Description

Creates a duplicate of this item and adds it this render queue.

NOTE: Duplicating an item whose status is "Done" sets the new item's status to "Queued".

Parameters

None.

Returns

RenderQueueItem object.

RenderQueueItem elapsedSeconds attribute

`app.project.renderQueue.item(index).elapsedSeconds`

Description

The number of seconds spent rendering this item.

Type

Integer, or null if item has not been rendered; read-only.

RenderQueueItem logType attribute

`app.project.renderQueue.item(`*`index`*`).logType`

Description

A log type for this item, indicating which events should be logged while this item is being rendered.

Type

A `LogType` enumerated value; (read/write). One of:

```
LogType.ERRORS_ONLY
LogType.ERRORS_AND_SETTINGS
LogType.ERRORS_AND_PER_FRAME_INFO
```

RenderQueueItem numOutputModules attribute

`app.project.renderQueue.item(`*`index`*`).numOutputModules`

Description

The total number of Output Modules assigned to this item.

Type

Integer; read-only.

RenderQueueItem onStatusChanged attribute

`app.project.renderQueue.item(`*`index`*`).onStatusChanged`

Description

The name of a callback function that is called whenever the value of the `RenderQueueItem.status` attribute changes. See “RenderQueueItem status attribute” on page 167.

You cannot make changes to render queue items or to the application while rendering is in progress or paused; you can, however, use this callback to pause or stop the rendering process. See “RenderQueue pauseRendering() method” on page 161 and “RenderQueue stopRendering() method” on page 162.

See also “Application onError attribute” on page 24.

Type

A function name string, or null if no function is assigned.

Example

```
function myStatusChanged() {
  alert(app.project.renderQueue.item(1).status)
}
```

```
app.project.renderQueue.item(1).onStatusChanged = myStatusChanged();
app.project.renderQueue.item(1).render = false; //changes status and shows dialog
```

RenderQueueItem outputModules attribute

`app.project.renderQueue.item(index).outputModules`

Description

The collection of Output Modules for the item.

Type

OMCollection object; read-only.

RenderQueueItem outputModule() method

`app.project.renderQueue.item(index).outputModule(index)`

Description

Gets an output module with the specified index position.

Parameters

`index`	The position index of the output module. An integer in the range [1..`numOutputModules`].

Returns

OutputModule object.

RenderQueueItem remove() method

`app.project.renderQueue.item(index).remove()`

Description

Removes this item from the render queue.

Parameters

None.

Returns

Nothing.

RenderQueueItem render attribute

`app.project.renderQueue.item(index).render`

Description

When true, the item will be rendered when the render queue is started. When set to true, the `RenderQueueItem.status` is set to `RQItemStatus.QUEUED`. When set to false, `status` is set to `RQItemStatus.UNQUEUED`.

Type

Boolean; read/write.

RenderQueueItem saveAsTemplate() method

`app.project.renderQueue.item(index).saveAsTemplate(name)`

Description

Saves the item’s current render settings as a new template with the specified name.

Parameters

`name`	A string containing the name of the new template.

Returns

Nothing.

RenderQueueItem skipFrames attribute

`app.project.renderQueue.item(index).skipFrames`

Description

The number of frames to skip when rendering this item. Use this to do rendering tests that are faster than a full render.

A value of 0 skip no frames, and results in regular rendering of all frames. A value of 1 skips every other frame. This is equivalent to "rendering on twos." Higher values skip a larger number of frames.

The total length of time remains unchanged. For example, if skip has a value of 1, a sequence output would have half the number of frames and in movie output, each frame would be double the duration.

Type

Integer in the range [0..99]. Read/write.

RenderQueueItem startTime attribute

`app.project.renderQueue.item(index).startTime`

Description

The day and time that this item started rendering.

Type

Date object, or null if the item has not started rendering; read-only.

RenderQueueItem status attribute

`app.project.renderQueue.item(index).status`

Description

The current render status of the item.

Type

An `RQItemStatus` enumerated value; read-only. One of:

`RQItemStatus.WILL_CONTINUE`	Rendering process has been paused.
`RQItemStatus.NEEDS_OUTPUT`	Item lacks a valid output path.
`RQItemStatus.UNQUEUED`	Item is listed in the Render Queue panel but composition is not ready to render.
`RQItemStatus.QUEUED`	Composition is ready to render.
`RQItemStatus.RENDERING`	Composition is rendering
`RQItemStatus.USER_STOPPED`	Rendering process was stopped by user or script.
`RQItemStatus.ERR_STOPPED`	Rendering process was stopped due to an error.
`RQItemStatus.DONE`	Rendering process for the item is complete.

RenderQueueItem templates attribute

`app.project.renderQueue.item(`*`index`*`).templates`

Description

The names of all Render Settings templates available for the item. See also "RenderQueueItem saveAsTemplate() method" on page 167.

Type

Array of strings; read-only.

RenderQueueItem timeSpanDuration attribute

`app.project.renderQueue.item(`*`index`*`).timeSpanDuration`

Description

The duration in seconds of the composition to be rendered. The duration is determined by subtracting the start time from the end time. Setting this value is the same as setting a custom end time in the Render Settings dialog box.

Type

Floating-point value; read/write.

RenderQueueItem timeSpanStart attribute

`app.project.renderQueue.item(`*`index`*`).timeSpanStart`

Description

The time in the composition, in seconds, at which rendering will begin. Setting this value is the same as setting a custom start time in the Render Settings dialog box.

Type

Floating-point value; read/write.

RQItemCollection object

`app.project.renderQueue.items`

Description

The RQItemCollection contains all of the render-queue items in a project, as shown in the Render Queue panel of the project. The collection provides access to the RenderQueueItem objects, and allows you to create them from compositions. The first RenderQueueItem object in the collection is at index position 1. See "RenderQueueItem object" on page 163

- RQItemCollection is a subclass of Collection. All methods and attributes of Collection are available when working with RQItemCollection. See "Collection object" on page 51.

Methods

Method	Reference	Description
`add()`	"RQItemCollection add() method" on page 169	Adds a composition to the Render Queue.

RQItemCollection add() method

`app.project.renderQueue.items.add(`*`comp`*`)`

Description

Adds a composition to the Render Queue, creating a RenderQueueItem.

Parameters

`comp`	The CompItem object for the composition to be added.

Returns

RenderQueueItem object.

Settings object

Description

The Settings object provides an easy way to manage settings for scripts. The settings are saved in the After Effects preferences file and are persistent between application sessions. Settings are identified by section and key within the file, and each key name is associated with a value. In the preferences file, section names are enclosed in brackets and quotation marks, and key names are listing in quotation marks below the section name. All values are strings.

You can create new settings with this object, as well as accessing existing settings.

Methods

Method	Reference	Description
`saveSetting()`	"Settings saveSetting() method" on page 171	Saves a default value for a setting.
`getSetting()`	"Settings getSetting() method" on page 170	Retrieves a setting value.
`haveSetting()`	"Settings haveSetting() method" on page 170	Reports whether a specified setting is assigned.

Settings getSetting() method

app.settings.getSetting(*sectionName, keyName*)

Description

Retrieves a scripting preferences item value from the preferences file.

Parameters

`sectionName`	A string containing the name of a settings section
`keyName`	A string containing the key name of the setting item.

Returns

String.

Example

If you have saved a setting named with the key name "Aligned Clone" in the "Eraser - Paint Settings" section, you can retrieve the value with this script:

```
var n = app.settings.getSetting("Eraser - Paint Settings", "Aligned Clone");
alert("The setting is " + n);
```

Settings haveSetting() method

app.settings.haveSetting(*sectionName, keyName*)

Description

Returns true if the specified scripting preferences item exists and has a value.

Parameters

sectionName	A string containing the name of a settings section
keyName	A string containing the key name of the setting item.

Returns

Boolean.

Settings saveSetting() method

app.settings.saveSetting(*sectionName, keyName, value*)

Description

Saves a default value for a scripting preferences item.

Parameters

sectionName	A string containing the name of a settings section
keyName	A string containing the key name of the setting item.
value	A string containing the new value.

Returns

Nothing.

Shape object

`app.project.item(index).layer(index).property(`*`(index)`*`).property("maskShape").value`

Description

The Shape object encapsulates information describing a shape in a shape layer, or the outline shape of a Mask. It is the value of the “Mask Path” AE properties, and of the "Path" AE property of a shape layer. Use the constructor, `new Shape()`, to create a new, empty Shape object, then set the attributes individually to define the shape.

A shape has a set of anchor points, or *vertices*, and a pair of direction handles, or *tangent vectors*, for each anchor point. A tangent vector (in a non-RotoBezier mask) determines the direction of the line that is drawn to or from an anchor point. There is one incoming tangent vector and one outgoing tangent vector associated with each vertex in the shape.

A tangent value is a pair of *x,y* coordinates specified relative to the associated vertex. For example, a tangent of [-1,-1] is located above and to the left of the vertex and has a 45 degree slope, regardless of the actual location of the vertex. The longer a handle is, the greater its influence; for example, an incoming shape segment stays closer to the vector for an `inTangent` of [-2,-2] than it does for an `inTangent` of [-1,-1], even though both of these come toward the vertex from the same direction.

If a shape is not closed, the `inTangent` for the first vertex and the `outTangent` for the final vertex are ignored. If the shape is closed, these two vectors specify the direction handles of the final connecting segment out of the final vertex and back into the first vertex.

RotoBezier masks calculate their tangents automatically. (See “MaskPropertyGroup rotoBezier attribute” on page 108.) If a shape is used in a RotoBezier mask, the tangent values are ignored. This means that, for RotoBezier masks, you can construct a shape by setting only the `vertices` attribute and setting both `inTangents` and `outTangents` to null. When you access the new shape, its tangent values are filled with the automatically calculated tangent values.

For closed mask shapes, variable-width mask feather points can exist anywhere along the mask path. Feather points are part of the Mask Path property. Reference a specific feather point by the number of the mask path segment (portion of the path between adjacent vertices) where it appears.

NOTE: The feather points on a mask are listed in an array in the order that they were created.

Example: Create a square mask

A square is a closed shape with 4 vertices. The `inTangents` and `outTangents` for connected straight-line segments are 0, the default, and do not need to be explicitly set.

```
var myShape = new Shape();
myShape.vertices = [[0,0], [0,100], [100,100], [100,0]];
myShape.closed = true;
```

Example: Create a “U” shaped mask

A "U" is an open shape with the same 4 vertices used in the square:

```
var myShape = new Shape();
myShape.vertices = [[0,0], [0,100], [100,100], [100,0]];
myShape.closed = false;
```

Example: Create an oval

An oval is a closed shape with 4 vertices and with `inTangent` and `outTangent` values:

```
var myShape = new Shape();
myShape.vertices = [[300,50],[200,150],[300,250],[400,150]];
myShape.inTangents = [[55.23,0],[0,-55.23],[-55.23,0],[0,55.23]];
myShape.outTangents = [[-55.23,0],[0,55.23],[55.23,0],[0,-55.23]];
myShape.closed = true;
```

Example: Create a square mask with two feather points

A large square mask with two feather points, one closer to the left end the second mask segment (off the bottom edge) with a radius of 30 pixels and the other one centered the third mask segment (off the right edge) with a larger radius of 100 pixels.

```
var myShape = new Shape();
myShape.vertices = [[100,100], [100,400], [400,400], [400,100]]; // segments drawn counterclockwise
myShape.closed = true;

myShape.featherSegLocs = [1,2]; // segments are numbered starting at 0, so second segment is 1
myShape.featherRelSegLocs = [0.15, 0.5]; // 0.15 is closer to the lower-left corner of the square
myShape.featherRadii = [30, 100]; // second feather point (on right-side segment) has a larger radius
```

Attributes

Attribute	Reference	Description
`closed`	"Shape closed attribute" on page 173	When true, the shape is a closed curve.
`vertices`	"Shape vertices attribute" on page 177	The anchor points of the shape.
`inTangents`	"Shape inTangents attribute" on page 176	The tangent vectors coming into the shape vertices.
`outTangents`	"Shape outTangents attribute" on page 176	The tangent vectors coming out of the shape vertices.
`featherSegLocs`	"Shape featherSegLocs attribute" on page 175	The mask path segment (sections of a mask path between vertices) containing each feather point.
`featherRelSegLocs`	"Shape featherRelSegLocs attribute" on page 175	The relative position of each feather point on its mask path segment.
`featherRadii`	"Shape featherRadii attribute" on page 174	The feather amount (radius) for each feather point.
`featherInterps`	"Shape featherInterps attribute" on page 174	The feather radius interpolation type for each feather point.
`featherTensions`	"Shape featherTensions attribute" on page 176	The feather tension at each feather point.
`featherTypes`	"Shape featherTypes attribute" on page 176	The direction (inner or outer) of each feather point.
`featherRelCornerAngles`	"Shape featherRelCornerAngles attribute" on page 174	The relative angle between the two normals on either side of a curved outer feather boundary at a corner on a mask path.

Shape closed attribute

`*shapeObject*.value.closed`

Description

When true, the first and last vertices are connected to form a closed curve. When false, the closing segment is not drawn.

Type

Boolean; read/write.

Shape featherInterps attribute

shapeObject.value.featherInterps

Description

An array containing each feather point's radius interpolation type (0 for non-Hold feather points, 1 for Hold feather points).

NOTE: Values are stored in the array in the order that feather points are created.

Type

Array of integers (0 or 1); read/write.

Shape featherRadii attribute

shapeObject.value.featherRadii

Description

An array containing each feather point's radius (feather amount); inner feather points have negative values.

NOTE: Values are stored in the array in the order that feather points are created.

Type

Array of floating-point values; read/write.

Shape featherRelCornerAngles attribute

shapeObject.value.featherRelCornerAngles

Description

An array containing each feather point's relative angle percentage between the two normals on either side of a curved outer feather boundary at a corner on a mask path. The angle value is 0% for feather points not at corners.

NOTE: Values are stored in the array in the order that feather points are created.

Type

Array of floating-point percentage values (0 to 100); read/write.

Shape featherRelSegLocs attribute

shapeObject.value.featherRelSegLocs

Description

An array containing each feather point's relative position, from 0 to 1, on its mask path segment (section of the mask path between vertices, numbered starting at 0).

NOTE: Values are stored in the array in the order that feather points are created.

To move a feather point to a different mask path segment, first change the `featherSegLocs` attribute value, then this attribute.

Type

Array of floating-point values (0 to 1); read/write.

Shape featherSegLocs attribute

shapeObject.value.featherSegLocs

Description

An array containing each feather point's mask path segment number (section of the mask path between vertices, numbered starting at 0).

NOTE: Values are stored in the array in the order that feather points are created.

Move a feather point to a different segment by changing both its segment number (this attribute) and, optionally, its `featherRelSegLocs` attribute value.

Type

Array of integers; read/write.

Example

```
// Assuming a rectangle closed mask (segments numbered 0-3) has 3 mask feather points,
// move all 3 feather points to the first mask segment.

// Get the Shape object for the mask, assumed here to be the first mask on the layer.
var my_maskShape = layer.mask(1).property("ADBE Mask Shape").value;

// Check where mask feather points are located.
// Note: They are stored in the order that they are added.
var where_are_myMaskFeatherPoints = my_maskShape.featherSegLocs;

// Move all 3 feather points to the first mask segment (numbered 0).
my_maskShape.featherSegLocs = [0, 0, 0];

// Update the mask path.
layer.mask(1).property("ADBE Mask Shape").setValue(my_maskShape);
```

Shape featherTensions attribute

`shapeObject.value.featherTensions`

Description

An array containing each feather point's tension amount, from 0 (0% tension) to 1 (100% tension).

NOTE: Values are stored in the array in the order that feather points are created.

Type

Array of floating-point values (0 to 1); read/write.

Shape featherTypes attribute

`shapeObject.value.featherTypes`

Description

An array containing each feather point's direction, either 0 (outer feather point) or 1 (inner feather point).

NOTE: You cannot change the direction of a feather point after it has been created.

NOTE: Values are stored in the array in the order that feather points are created.

Type

Array of integers (0 or 1); read/write.

Shape inTangents attribute

`shapeObject.value.inTangents`

Description

The incoming tangent vectors, or direction handles, associated with the vertices of the shape. Specify each vector as an array of two floating-point values, and collect the vectors into an array the same length as the `vertices` array.

Each tangent value defaults to [0,0]. When the mask shape is not RotoBezier, this results in a straight line segment.

If the shape is in a RotoBezier mask, all tangent values are ignored and the tangents are automatically calculated.

Type

Array of floating-point pair arrays; read/write.

Shape outTangents attribute

`shapeObject.value.outTangents`

Description

The outgoing tangent vectors, or direction handles, associated with the vertices of the shape. Specify each vector as an array of two floating-point values, and collect the vectors into an array the same length as the `vertices` array.

Each tangent value defaults to [0,0]. When the mask shape is not RotoBezier, this results in a straight line segment.

If the shape is in a RotoBezier mask, all tangent values are ignored and the tangents are automatically calculated.

Type

Array of floating-point pair arrays; read/write.

Shape vertices attribute

shapeObject.value.vertices

Description

The anchor points of the shape. Specify each point as an array of two floating-point values, and collect the point pairs into an array for the complete set of points. For example:

```
myShape.vertices = [[0,0], [0,1], [1,1], [1,0]];
```

Type

Array of floating-point pair arrays; read/write.

ShapeLayer object

`app.project.item(index).layer(index)`

Description

The ShapeLayer object represents a shape layer within a composition. Create it using the LayerCollection object's `addShape()` method; see "LayerCollection addShape() method" on page 97. It can be accessed in an item's layer collection either by index number or by a name string.

- ShapeLayer is a subclass of AVLayer, which is a subclass of Layer. All methods and attributes of AVLayer and Layer are available when working with ShapeLayer. See "Layer object" on page 86 and "AVLayer object" on page 38.

SolidSource object

app.project.item(index).mainSource
app.project.item(index).proxySource

Description

The SolidSource object represents a solid-color footage source.

- SolidSource is a subclass of FootageSource. All methods and attributes of FootageSource, in addition to those listed below, are available when working with SolidSource. See “FootageSource object” on page 69.

Attributes

Attribute	Reference	Description
color	“SolidSource color attribute” on page 179	The color of the solid.

SolidSource color attribute

solidSource.color

Description

The color of the solid, expressed as red, green, and blue values.

Type

Array of three floating-point values, [R, G, B], in the range [0.0..1.0]; read/write.

System object

`system`

Description

The System object provides access to attributes found on the user's system, such as the user name and the name and version of the operating system. It is available through the `system` global variable.

Example

```
alert ("Your OS is " + system.osName + " running version " + system.osVersion);
confirm("You are: " + system.userName + " running on " + system.machineName + ".");
```

Attributes

Attribute	Reference	Description
`userName`	"System userName attribute" on page 181	The current user name.
`machineName`	"System machineName attribute" on page 181	The name of the host computer.
`osName`	"System osName attribute" on page 181	The name of the operating system.
`osVersion`	"System osVersion attribute" on page 181	The version of the operating system.

Methods

Method	Reference	Description
`callSystem()`	"System callSystem() method" on page 180	Execute's a command on the system's command line.

System callSystem() method

`system.callSystem (cmdLineToExecute);`

Description

Executes a system command, as if you had typed it on the operating system's command line. Returns whatever the system outputs in response to the command, if anything.

In Windows, you can invoke commands using the `/c` switch for the `cmd.exe` command, passing the command to run in escaped quotes (`\"...\"`). For example, the following retrieves the current time and displays it to the user:

```
var timeStr = system.callSystem("cmd.exe /c \"time /t\"");
alert("Current time is " + timeStr);
```

Parameters

`cmdLineToExecute`	A string containing the command and its parameters.

Returns

The output from the command.

System machineName attribute

`system.machineName`

Description

The name of the computer on which After Effects is running.

Type

String; read-only.

System osName attribute

`system.osName`

Description

The name of the operating system on which After Effects is running.

NOTE: As of Windows 7, this attribute returns a blank value. Use `$.os` *instead.*

Type

String; read-only.

System osVersion attribute

`system.osVersion`

Description

The version of the current local operating system.

Type

String; read-only.

System userName attribute

`system.userName`

Description

The name of the user currently logged on to the system.

Type

String; read-only.

TextDocument object

```
new TextDocument(docText)
app.project.item(index).layer(index).property("Source Text").value
```

Description

The TextDocument object stores a value for a TextLayer's Source Text property. Create it with the constructor, passing the string to be encapsulated.

Examples

This sets a value of some source text and displays an alert showing the new value:

```
var myTextDocument = new TextDocument("Happy Cake");
myTextLayer.property("Source Text").setValue(myTextDocument);
alert(myTextLayer.property("Source Text").value);
```

This sets keyframe values for text that show different words over time:

```
var textProp = myTextLayer.property("Source Text");
textProp.setValueAtTime(0, new TextDocument("Happy"));
textProp.setValueAtTime(.33, new TextDocument("cake"));
textProp.setValueAtTime(.66, new TextDocument("is"));
textProp.setValueAtTime(1, new TextDocument("yummy!"));
```

This sets various character and paragraph settings for some text:

```
var textProp = myTextLayer.property("Source Text");
var textDocument = textProp.value;
myString = "Happy holidays!";
textDocument.resetCharStyle();
textDocument.fontSize = 60;
textDocument.fillColor = [1, 0, 0];
textDocument.strokeColor = [0, 1, 0];
textDocument.strokeWidth = 2;
textDocument.font = "TimesNewRomanPSMT";
textDocument.strokeOverFill = true;
textDocument.applyStroke = true;
textDocument.applyFill = true;
textDocument.text = myString;
textDocument.justification = ParagraphJustification.CENTER_JUSTIFY;
textDocument.tracking = 50;
textProp.setValue(textDocument);
```

Attributes

Attribute	Reference	Description
`text`	"TextDocument text attribute" on page 187	The text layer's Source Text value.
`font`	"TextDocument font attribute" on page 184	The text layer's font specified by its PostScript name.
`fontSize`	"TextDocument fontSize attribute" on page 185	The text layer's font size in pixels.
`applyFill`	"TextDocument applyFill attribute" on page 183	When true, the text layer shows a fill.

Attribute	Reference	Description
applyStroke	“TextDocument applyStroke attribute” on page 183	When true, the text layer shows a stroke.
fillColor	“TextDocument fillColor attribute” on page 184	The text layer’s fill color.
strokeColor	“TextDocument strokeColor attribute” on page 186	The text layer’s stroke color.
strokeOverFill	“TextDocument strokeOverFill attribute” on page 186	Indicates the rendering order for the fill and stroke of a text layer.
strokeWidth	“TextDocument strokeWidth attribute” on page 186	The text layer’s stroke thickness.
justification	“TextDocument justification attribute” on page 185	The paragraph justification for the text layer.
tracking	“TextDocument tracking attribute” on page 187	The text layer’s spacing between characters.
pointText	“TextDocument pointText attribute” on page 185	When true, the text layer is point (unbounded) text.
boxText	“TextDocument boxText attribute” on page 184	When true, the text layer is paragraph (bounded) text.
boxTextSize	“TextDocument boxTextSize attribute” on page 184	For box text, the pixel dimensions for the text bounds.

Methods

Method	Reference	Description
resetCharStyle().	“TextDocument resetCharStyle() method” on page 185	Restores the default character settings in the Character panel.
resetParagraphStyle()	“TextDocument resetParagraphStyle() method” on page 186	Restores the default paragraph settings in the Paragraph panel.

TextDocument applyFill attribute

textDocument.applyFill

Description

When true, the text layer shows a fill. Access the fillColor attribute for the actual color. When false, only a stroke is shown.

Type

Boolean; read/write.

TextDocument applyStroke attribute

textDocument.applyStroke

Description

When true, the text layer shows a stroke. Access the strokeColor attribute for the actual color and strokeWidth for its thickness. When false, only a fill is shown.

Type

Boolean; read/write.

TextDocument boxText attribute

textDocument.boxText

Description

True if a text layer is a layer of paragraph (bounded) text; otherwise false.

Type

Boolean; read-only.

TextDocument boxTextSize attribute

textDocument.boxTextSize

Description

The size of a paragraph (box) text layer as a [width, height] array of pixel dimensions.

Type

Array of two integers (minimum value of 1); read/write.

TextDocument fillColor attribute

textDocument.fillColor

Description

The text layer's fill color, as an array of [r, g, b] floating-point values. For example, in an 8-bpc project, a red value of 255 would be 1.0, and in a 32-bpc project, an overbright blue value can be something like 3.2.

NOTE: If the text layer has different fill color settings for each character, this attribute returns the setting for the first character. Also, if you change the value, it resets all characters in the text layer to the specified setting.

Type

Array [r, g, b] of floating-point values; read/write.

TextDocument font attribute

textDocument.font

Description

The text layer's font specified by its PostScript name.

NOTE: If the text layer has different font settings for each character, this attribute returns the setting for the first character. Also, if you change the value, it resets all characters in the text layer to the specified setting.

Type

String; read/write.

TextDocument fontSize attribute

textDocument.fontSize

Description

The text layer's font size in pixels.

NOTE: If the text layer has different font size settings for each character, this attribute returns the setting for the first character. Also, if you change the value, it resets all characters in the text layer to the specified setting.

Type

Floating-point value (0.1 to 1296, inclusive); read/write.

TextDocument justification attribute

textDocument.justification

Description

The paragraph justification for the text layer.

Type

A ParagraphJustification enumerated value; read-only. One of:

```
ParagraphJustification.LEFT_JUSTIFY
ParagraphJustification.RIGHT_JUSTIFY
ParagraphJustification.CENTER_JUSTIFY
ParagraphJustification.FULL_JUSTIFY_LASTLINE_LEFT
ParagraphJustification.FULL_JUSTIFY_LASTLINE_RIGHT
ParagraphJustification.FULL_JUSTIFY_LASTLINE_CENTER
ParagraphJustification.FULL_JUSTIFY_LASTLINE_FULL
```

TextDocument pointText attribute

textDocument.pointText

Description

True if a text layer is a layer of point (unbounded) text; otherwise false.

Type

Boolean; read-only.

TextDocument resetCharStyle() method

textDocument.resetCharStyle()

Description

Restores the default text character characteristics in the Character panel.

Parameters

None.

Returns

Nothing.

TextDocument resetParagraphStyle() method

textDocument.resetParagraphStyle()

Description

Restores the default text paragraph characteristics in the Paragraph panel.

Parameters

None.

Returns

Nothing.

TextDocument strokeColor attribute

textDocument.strokeColor

Description

The text layer’s stroke color, as an array of [r, g, b] floating-point values. For example, in an 8-bpc project, a red value of 255 would be 1.0, and in a 32-bpc project, an overbright blue value can be something like 3.2.

NOTE: If the text layer has different stroke color settings for each character, this attribute returns the setting for the first character. Also, if you change the value, it resets all characters in the text layer to the specified setting.

Type

Array [r, g, b] of floating-point values; read/write.

TextDocument strokeOverFill attribute

textDocument.strokeOverFill

Description

Indicates the rendering order for the fill and stroke of a text layer. When true, the stroke appears over the fill.

NOTE: If the text layer has different fill/stroke rendering order settings for each character, this attribute returns the setting for the first character. Also, if you change the value, it resets all characters in the text layer to the specified setting.

Type

Boolean; read/write.

TextDocument strokeWidth attribute

textDocument.strokeWidth

Description

The text layer’s stroke thickness in pixels.

NOTE: If the text layer has different stroke width settings for each character, this attribute returns the setting for the first character. Also, if you change the value, it resets all characters in the text layer to the specified setting.

Type

Floating-point value (0 to 1000, inclusive); read/write.

TextDocument text attribute

textDocument.text

Description

The text value for the text layer's Source Text property.

Type

String; read/write.

TextDocument tracking attribute

textDocument.tracking

Description

The text layer's spacing between characters.

NOTE: If the text layer has different tracking settings for each character, this attribute returns the setting for the first character. Also, if you change the value, it resets all characters in the text layer to the specified setting.

Type

Floating-point value; read/write.

TextLayer object

`app.project.item(index).layer(index)`

Description

The TextLayer object represents a text layer within a composition. Create it using the LayerCollection object's `addText` method; see "LayerCollection addText() method" on page 98. It can be accessed in an item's layer collection either by index number or by a name string.

- TextLayer is a subclass of AVLayer, which is a subclass of Layer. All methods and attributes of AVLayer and Layer are available when working with TextLayer. See "Layer object" on page 86 and "AVLayer object" on page 38.

AE Properties

TextLayer defines no additional attributes, but has the following AE properties and property groups, in addition to those inherited from AVLayer:

```
Text
    Source Text
    Path Options
        Path
        Reverse Path
        Perpendicular To Path
        Force Alignment
        First Margin
        Last Margin
    More Options
        Anchor Point Grouping
        Grouping Alignment
        Fill & Stroke
        Inter-Character Blending
    Animators
```

Unused Properties and Attributes

The `Time Remap` and `Motion Trackers` properties, inherited from AVLayer, are not applicable to text layers, and their related AVLayer attributes are not used:

```
canSetTimeRemapEnabled
timeRemapEnabled
trackMatteType
isTrackMatte
hasTrackMatte
```

Viewer object

`app.activeViewer`

Description

The Viewer object represents a Composition, Layer, or Footage panel.

Example

This maximizes the active viewer panel, and displays its type if it contains a composition:

```
var activeViewer = app.activeViewer;
activeViewer.maximized = true;
if (activeViewer.type == ViewerType.VIEWER_COMPOSITION)
    alert("Composition panel is active.");
```

Attributes

Attribute	Reference	Description
`type`	"Viewer type attribute" on page 190	The type of content in the viewer.
`active`	"Viewer active attribute" on page 189	When true, the viewer is focused.
`maximized`	"Viewer maximized attribute" on page 189	When true, the viewer is at its maximized size.

Methods

Method	Reference	Description
`setActive()`.	"Viewer setActive() method" on page 190	Moves the viewer to front and places focus on it.

Viewer active attribute

viewer.active

Description

When true, indicates if the viewer panel is focused, and thereby frontmost.

Type

Boolean; read-only.

Viewer maximized attribute

viewer.maximized

Description

When true, indicates if the viewer panel is at its maximized size.

Type

Boolean; read/write.

Viewer setActive() method

viewer.setActive()

Description

Moves the viewer panel to the front and places focus on it, making it active. Calling this method will set the viewer's `active` attribute to true.

Parameters

None.

Returns

Boolean indicating if the viewer panel was made active.

Viewer type attribute

viewer.type

Description

The content in the viewer panel.

Type

A `ViewerType` enumerated value; read-only. One of:

```
ViewerType.VIEWER_COMPOSITION
ViewerType.VIEWER_LAYER
ViewerType.VIEWER_FOOTAGE
```

Examples

This section describes some of the sample scripts available from the File > Scripts menu, giving an overview of what they do and a description of how they work.

This set of examples is by no means exhaustive, but it does demonstrate some of scripting's more complex features in action. It also shows some typical programming constructions from JavaScript that apply to scripting.

For more examples from Adobe and from other After Effects users, visit Adobe Exchange at http://www.adobe.com/go/exchange/, and choose the Scripts category in the Adobe After Effects section.

New render locations

This script, `Change Render Locations.jsx`, allows the user to select queued items in the render queue and assign a new render destination for them.

This script does the following:

- Prompts the user for a new folder to use as a render destination.
- Checks that the user entered a new location (and didn't cancel), then creates a loop for each selected render queue item, and for each output module in it.
- If an item is queued, gives the current render location a new name and location, and displays an alert stating the new file path.

Smart import

This script, `Smart Import.jsx`, allows the user to import the full, nested contents of a folder just by selecting it. It attempts to detect whether each item is a still, moving footage, or an image sequence. The user still has to make other choices in dialog boxes, such as which layer of a multi-layer image (such as a PSD file) to import.

This script does the following:

- Prompts the user for a folder whose contents are to be imported, and checks that the user chooses a folder rather than cancelling.
- Defines a function, `processFolder()`, to import each of the files in the chosen folder, which uses several helper functions.
- Defines a helper function, `testForSequence()`, to test whether a given file is part of a sequence. This uses regular expressions, which are a special type of JavaScript designed to reduce the number of steps required to evaluate a string.

  The first one tests for the presence of sequential numbers anywhere in the file name, followed by another making certain that the sequential files aren't of a type that can't be imported as a sequence (moving image files). The function then checks adjacent files to see if a sequence exists, stopping after we've evaluated ten files to save processing time.

If no match is found for a number string, assumes there is no image sequence and checks for an array consisting of the matched string and its location within the file name.

If all files are part of a numbered sequence, assumes a sequence and returns the first file of that sequence.

- Defines a helper function to pop up error dialog boxes if there is a problem with any file we are attempting to import.
- Defines a helper function to actually import any image sequence discovered using `testForSequence()`. There is an option for forcing alphabetical order in sequences, which is commented out in the script as written. If you want to force alphabetical order, uncomment the line `importOptions.forceAlphabetical = true`.
- Calls the main function, `processFolder()`.

Render and e-mail

This script, `Render and Email.jsx`, renders all queued items in an open project and sends an e-mail report to indicate when the render has completed. It makes use of two other scripts that follow, `email_methods.jsx` (to send the e-mail properly) and `Change Email Settings.jsx` (which establishes the sender, recipient, and e-mail server); these two scripts are in the *(support)* subfolder of the *Scripts* folder on disk.

This script does the following:

- Establishes conditions under which the script will run. An open project with at least one item queued is required.
- Checks whether e-mail settings are already saved in the preferences. If not, run the `Change Email Settings.jsx` script (via File > Scripts > Run Script File), which prompts the user for the mail gateway and sender and recipient addresses. (If there are saved settings that you need to change, you can always run the script to make new settings that overwrite the existing ones.)
- Render the items in the render queue.
- When rendering is complete, creates a text string for the e-mail message that contains the start time of the render, the render time of each item in the queue, and the total render time.
- E-mails the message, using the settings (such as the server) from the `email_methods.jsx` script
- Displays an error if for any reason it is unable to send the mail.

A helper script, `email_methods.jsx`, creates an e-mail object, using the ExtendScript Socket object. For details of that utility, see the *Creative Suite 6 JavaScript Tools Guide*.

Another helper script, `email_setup.jsx`, prompts the user for the server name, e-mail sender, and e-mail recipient that are saved as Settings. You can run this script as standalone any time you want to change the settings. This script is a good example of how to create settings that are saved in preferences for the sole use of scripting (as opposed to altering existing After Effects preferences settings).

Convert selected properties to markers

This script, `Convert Selected Properties to Markers.jsx`, goes through the properties in each layer that are currently selected in the Timeline panel, and converts the value of each property at each frame time to a Flash Video event cue point in a marker.

This script adds a layer-time marker on the layer at the same time as each keyframe for each selected property. Each marker is associated with an event-type Flash Video cue point, and the cue point is given a parameter whose name is the name of the property and whose value is the property's value at that time. If the selected property has an expression, a marker is created for each frame, with the values sampled at each frame.

Note: This script does not convert properties that have complex value types, such as the Path property for a paint stroke, the Curves property of a Curve effect, or a gradient property.

When you render the composition as Flash video, all markers that contain cue-point data are converted to Flash Video cue points.