After Effects Scripting Guide

Release 22.3.0

Adobe Systems Incorporated

Apr 25, 2024

INTRODUCTION

CHAPTER

ONE

OVERVIEW

1.1 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.

Although both the After Effects expressions language and the After Effects ExtendScript scripting language is 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.

1.2 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 pathname 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 provide s a common scripting environment for all Adobe applications, and allows inter-application communication through scripts.

1.3 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.

1.4 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.

1.5 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.

1.6 Loading and running scripts

1.6.1 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.

1.6.2 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.

1.6.3 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"
```

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

The 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
```

Note: This documentation is incorrect, the correct invocation in this instance is `DoScriptFile`

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
```

1.6.5 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* 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

1.6.6 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);
```

1.6.7 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.

CHAPTER

TWO

CHANGELOG

What's new and changed for scripting?

2.1 After Effects 24.4 Beta build 25 (March 2024)

- Added: *ThreeDModelLayer object*

2.2 After Effects 24.4 Beta build 24 (March 2024)

- Added: *FontsObject.favoriteFontFamilyList*
- Added: *FontsObject.fontsDuplicateByPostScriptName*
- Added: *FontsObject.freezeSyncSubstitutedFonts*
- Added: *FontsObject.mruFontFamilyList*
- Added: *FontsObject.substitutedFontReplacementMatchPolicy*
- Added: *FontsObject.pollForAndPushNonSystemFontFoldersChanges()*

2.3 After Effects 24.4 Beta build 6 (March 2024)

- **Scripting methods and attributes added or changed**
  - Changed: *Project.replaceFont()*
  - Changed: *Project.usedFonts*

2.4 After Effects 24.3 Beta build 25 (February 2024)

- **Scripting methods and attributes added or changed**
  - Changed: *ParagraphRange object* (all text attributes removed)
  - Added: *ComposedLineRange object*
  - Added: *ParagraphRange.characterRange()*
  - Added: *TextDocument.composedLineCharacterIndexesAt()*
  - Added: *TextDocument.composedLineCount*
  - Added: *TextDocument.composedLineRange()*

2.5 After Effects 24.3 Beta build 20 (February 2024)

- **Scripting methods and attributes added or changed**
  - Added: *TextDocument.boxAutoFitPolicy*
  - Added: *TextDocument.boxFirstBaselineAlignment*
  - Added: *TextDocument.boxFirstBaselineAlignmentMinimum*
  - Added: *TextDocument.boxInsetSpacing*
  - Added: *TextDocument.boxOverflow*
  - Added: *TextDocument.boxVerticalAlignment*

2.6 After Effects 24.2 (February 2024)

- **Scripting methods and attributes added or changed**
  - Added: *LayerCollection.addVerticalText()*
  - Added: *LayerCollection.addVerticalBoxText()*
  - Added: *TextDocument.lineOrientation*
  - Added: *FontsObject.fontServerRevision*
  - Added: *FontsObject.getFontByID()*
  - Added: *FontObject.fontID*

2.7 After Effects 24.2 Beta build 17 (November 2023)

- **Scripting methods and attributes added**
  - Added: *CharacterRange object*
  - Added: *ParagraphRange object*
  - Added: *TextDocument.characterRange()*
  - Added: *TextDocument.paragraphRange()*
  - Added: *TextDocument.paragraphCount*
  - Added: *TextDocument.paragraphCharacterIndexesAt()*

2.8 After Effects 24.0 Beta build 37 (August 2023)

- **Scripting methods and attributes added**
  - Added: *Project.usedFonts*
  - Added: *Project.replaceFont()*

2.9 After Effects 24.0 (October 2023)

- **Scripting methods and attributes added**
  - Added: *getEnumAsString()*
  - Added: *app.fonts*
  - Added: *Fonts object*
  - Added: *FontsObject.allFonts*
  - Added: *FontsObject.fontsWithDefaultDesignAxes*
  - Added: *FontsObject.getFontsByFamilyNameAndStyleName()*
  - Added: *FontsObject.getFontsByPostScriptName()*
  - Added: *FontsObject.missingOrSubstitutedFonts*
  - Added: *Font object*
  - Added: *FontObject.designAxesData*
  - Added: *FontObject.designVector*
  - Added: *FontObject.familyPrefix*
  - Added: *FontObject.hasDesignAxes*
  - Added: *FontObject.hasSameDict()*
  - Added: *FontObject.postScriptNameForDesignVector()*
  - Added: *FontObject.familyName*
  - Added: *FontObject.fullName*
  - Added: *FontObject.isFromAdobeFonts*

  - Added: *FontObject.isSubstitute*
  - Added: *FontObject.location*
  - Added: *FontObject.nativeFamilyName*
  - Added: *FontObject.nativeFullName*
  - Added: *FontObject.nativeStyleName*
  - Added: *FontObject.postScriptName*
  - Added: *FontObject.styleName*
  - Added: *FontObject.technology*
  - Added: *FontObject.type*
  - Added: *FontObject.version*
  - Added: *FontObject.writingScripts*
  - Added: *TextDocument.autoHyphenate*
  - Added: *TextDocument.autoKernType*
  - Added: *TextDocument.baselineDirection*
  - Added: *TextDocument.composerEngine*
  - Added: *TextDocument.digitSet*
  - Added: *TextDocument.direction*
  - Added: *TextDocument.endIndent*
  - Added: *TextDocument.everyLineComposer*
  - Added: *TextDocument.firstLineIndent*
  - Added: *TextDocument.fontBaselineOption*
  - Added: *TextDocument.fontCapsOption*
  - Added: *TextDocument.fontObject*
  - Added: *TextDocument.hangingRoman*
  - Added: *TextDocument.kerning*
  - Added: *TextDocument.leadingType*
  - Added: *TextDocument.ligature*
  - Added: *TextDocument.lineJoinType*
  - Added: *TextDocument.noBreak*
  - Added: *TextDocument.spaceAfter*
  - Added: *TextDocument.spaceBefore*
  - Added: *TextDocument.startIndent*

- **Scripting attributes updated**
  - Updated: *TextDocument.fauxBold*
  - Updated: *TextDocument.fauxItalic*
  - Updated: *TextDocument.justification*

2.10 After Effects 23.0 (October 2022)

- **Scripting methods and attributes added**
  - Added: *AVLayer.setTrackMatte()*
  - Added: *AVLayer.removeTrackMatte()*
  - Added: *AVLayer.trackMatteLayer*
- **Scripting attributes updated**
  - Updated: *AVLayer.trackMatteType*
  - Updated: *AVLayer.isTrackMatte*
  - Updated: *AVLayer.hasTrackMatte*

2.11 After Effects 22.6 (August 2022)

- **Scripting methods added**
  - Added: *Property.keyLabel()*
  - Added: *Property.setLabelAtKey()*

2.12 After Effects 22.3 (April 2022)

- **Scripting methods added**
  - Added: *Layer.doSceneEditDetection()*

2.13 After Effects 22.0 (October 2021)

- **Scripting methods added**
  - Added: *Layer.id*
  - Added: *Project.layerByID()*
  - Added: *Property.essentialPropertySource*
- **Scripting Access to Render Queue Notifications**
  - Added: *RenderQueue.queueNotify*
  - Added: *RenderQueueItem.queueItemNotify*
- **Scripting Access to Multi-Frame Rendering, Maximum CPU Percentage Overrides**
  - Added: *app.setMultiFrameRenderingConfig()*

2.14 After Effects 18.0 (March 2021)

- **Scripting methods and attributes to support Media Replacement**
  - Added: *AVItem.isMediaReplacementCompatible*
  - Added: *AVLayer.addToMotionGraphicsTemplate()*
  - Added: *AVLayer.addToMotionGraphicsTemplateAs()*
  - Added: *AVLayer.canAddToMotionGraphicsTemplate()*
  - Added: *Property.alternateSource*
  - Added: *Property.canSetAlternateSource*
  - Added: *Property.setAlternateSource()*
  - Added relevant *match names*
- Added *match name for Essential Properties* property group.

2.15 After Effects 17.1.1 (May 2020)

- **Scripting access to Shape Layer Stroke Taper, Stroke Waves, Offset Paths Copies, Offset Path Copy Offset**
  - Added relevant *match names*
- **Fixed an issue to allow negative values for *CompItem.displayStartTime*:**
  - Added *CompItem.displayStartFrame*
  - Now matches the valid range allowed when setting the Start Timecode in the Composition Settings Dialog (-3:00:00:00 to 23:59:00:00).

2.16 After Effects 17.0.1 (November 2019)

- **Scripted creation and modification of Dropdown Menu Control items:**
  - Added: *Property.isDropdownEffect*
  - Added: *Property.setPropertyParameters()*

2.17 After Effects 16.1

- **Scripting access to *ViewOptions object* guide and ruler booleans:**
  - Added: *ViewOptions.guidesLocked*
  - Added: *ViewOptions.guidesSnap*
  - Added: *ViewOptions.guidesVisibility*
  - Added: *ViewOptions.rulers*
- **Scripting access to add, remove, and set existing guides:**
  - Added: *Item.addGuide()*
  - Added: *Item.removeGuide()*
  - Added: *Item.setGuide()*
- **Scripting access to additional EGP property attributes:**
  - Added: *CompItem.motionGraphicsTemplateControllerCount*
  - Added: *CompItem.getMotionGraphicsTemplateControllerName()*
  - Added: *CompItem.setMotionGraphicsControllerName()*
  - Added: *Property.addToMotionGraphicsTemplateAs()*

2.18 After Effects 16.0 (October 2018)

- **Scripting access to marker label and protectedRegion attributes:**
  - Added: *MarkerValue.label*
  - Added: *MarkerValue.protectedRegion*
- **Scripting access to additional project color management settings:**
  - Added: *Project.workingSpace*
  - Added: *Project.workingGamma*
  - Added: *Project.listColorProfiles()*
  - Added: *Project.linearizeWorkingSpace*
  - Added: *Project.compensateForSceneReferredProfiles*
- **Scripting access to the expression engine attribute:**
  - Added: *Project.expressionEngine*
- Added project method *Project.setDefaultImportFolder()*, which sets the folder that will be shown in the file import dialog.
- Added app property *app.disableRendering*, which disables rendering via the same mechanism as the Caps Lock key.

2.19 After Effects 15.1 (April 2018)

- *Project.autoFixExpressions()* will now fix expression name references in single quotes (ex., ('Effect Name')), as well as double quotes.
- Fixes *CompItem.exportAsMotionGraphicsTemplate()* not returning a boolean as expected

---

2.20 After Effects 15.0 (October 2017)

- **Scripting Access to motion graphics templates**
  - Added: *CompItem.motionGraphicsTemplateName*
  - Added: *CompItem.exportAsMotionGraphicsTemplate()*
  - Added: *CompItem.openInEssentialGraphics()*
  - Added: *Property.addToMotionGraphicsTemplate()*
  - Added: *Property.canAddToMotionGraphicsTemplate()*

2.21 After Effects 14.2.1 (CC 2017.2) (June 2017)

- Buttons in ScriptUI panels have been reverted to the rectangular appearance seen in After Effects 14.1 and previous releases.
- The *AVItem.setProxyToNone()* scripting method no longer fails with an error message, "After Effects error: AEGP trying to add invalid footage".
- The *System.callSystem()* scripting method now waits for all tasks called by the command to complete, instead of failing when the command takes a long time to complete.

2.22 After Effects 14.2 (CC 2017.1) (April 2017)

- **Scripting Access to text leading**
  - Added: *TextDocument.leading*
- **Scripting Access to Team Projects (Beta)**
  - Added: *Project.newTeamProject()*
  - Added: *Project.openTeamProject()*
  - Added: *Project.shareTeamProject()*
  - Added: *Project.syncTeamProject()*
  - Added: *Project.closeTeamProject()*
  - Added: *Project.convertTeamProjectToProject()*

  - Added: *Project.listTeamProjects()*
  - Added: *Project.isTeamProjectOpen()*
  - Added: *Project.isAnyTeamProjectOpen()*
  - Added: *Project.isTeamProjectEnabled()*
  - Added: *Project.isLoggedInToTeamProject()*
  - Added: *Project.isSyncCommandEnabled()*
  - Added: *Project.isShareCommandEnabled()*
  - Added: *Project.isResolveCommandEnabled()*
  - Added: *Project.resolveConflict()*
- Drop-down menus in ScriptUI panels are no longer clipped on HiDPI displays on Windows.
- The appearance of buttons, sliders, disclosure triangles ("twirly arrow"), scroll bar, progress bar, radio buttons, and checkboxes in ScriptUI embedded panels have been updated to match the appearance of After Effects native controls.
- After Effects no longer crashes when the *AVLayer.compPointToSource()* scripting method is used with a 3D text layer.
- The match name of the Fast Box Blur effect is "ADBE Box Blur2". The older match name "ADBE Box Blur" will continue to work: when used to add the effect, "ADBE Box Blur" will apply the Fast Box Blur effect, but with the older name "Box Blur"; the Iterations parameter will be set to the new default of 3.

2.23 After Effects 14.0 (CC 2017) (November 2016)

- **Scripting Access to Tools**
  - Added: *Project.toolType*
- **Scripting Access to Composition Markers**
  - Added: *CompItem.markerProperty*
- **Scripting Access to Queue in AME**
  - Added: *RenderQueue.queueInAME()*
- **Scripting Access to Available GPU Acceleration Options**
  - Added: *app.availableGPUAccelTypes*

2.24 After Effects 13.8 (CC 2015.3) (June 2016)

- **Enable GPU effect rendering via scripting**
  - Added: *Project.gpuAccelType*
- New Gaussian Blur effect added w/ matchname `ADBE Gaussian Blur 2`

2.25 After Effects 13.6 (CC 2015) (November 2015)

- **Scripting access to text baselines**
  - Added: *baselineLocs*
- **New scripting method to generate random numbers**
  - Added: *generateRandomNumber()*
- Using the *copyToComp()* scripting method no longer causes After Effects to crash when the layer has a parent.
- The *valueAtTime()* scripting method now waits for time-intensive expressions, like `sampleImage`, to finish evaluating before it returns the result.
- ScriptUI panels now display and resize correctly on high-DPI displays on Windows.
- After Effects no longer crashes when you click OK or Cancel buttons in a scriptUI dialog with tabbed panels.

2.26 After Effects 13.2 (CC 2014.2) (December 2014)

- **Scripting improvements for text layers (read-only)**
  - **Returns boolean value:**
    - Added: *fauxBold*
    - Added: *fauxItalic*
    - Added: *allCaps*
    - Added: *smallCaps*
    - Added: *superscript*
    - Added: *subscript*
  - **Returns float:**
    - Added: *verticalScale*
    - Added: *horizontalScale*
    - Added: *baselineShift*
    - Added: *tsume*
  - **Returns array of ([X,Y]) position coordinates (paragraph text layers only):**
    - Added: *boxTextPos*

- **Layer space / comp space conversion:**
  - Added: *sourcePointToComp()*
  - Added: *compPointToSource()*

2.27 After Effects 13.1 (CC 2014.1) (September 2014)

- **Scripting improvements for text layers (read-only)**
  - **returns string:**
    - Added: *fontLocation*
    - Added: *fontStyle*
    - Added: *fontFamily*
- "Use Legacy UI" toggle implemented

2.28 After Effects 13.0 (CC 2014) (June 2014)

- **Scripting access to render settings and output module settings**
  - Added: RenderQueueItem object *getSetting*, *setSetting* methods
  - Added: RenderQueueItem object *getSettings*, *setSettings* methods
  - Added: OutputModule object *getSetting*, *setSetting* methods
  - Added: OutputModule object *getSettings*, *setSettings* methods
- Fetch project item by id: *Project.itemByID()*
- CEP panels implemented

2.29 After Effects 12.0 (CC) (June 2013)

- **Access to effect's internal version string**
  - Added: Application effects object's version attribute, see *app.effects*
- **Ability to get and set preview mode**
  - Added: *ViewOptions.fastPreview*
- Access to layer sampling method (see *samplingQuality*)
- Changed preference and settings methods (see *Settings object*)
- ScriptUI is now based on the same controls as the main application.

2.30 After Effects 11.0 (CS6) (April 2012)

- **Added: Access to *Viewer object* object and controls**
  - Added: *app.activeViewer*
  - Added: *AVLayer.openInViewer()* to open a layer in the layer viewer
  - Added: *CompItem.openInViewer()* to open a composition in the composition viewer
  - Added: *FootageItem.openInViewer()* to open a footage item in the footage viewer
- Added: *Property.canSetExpression*
- Added: *AVLayer.environmentLayer*
- Added: *MaskPropertyGroup.maskFeatherFalloff*
- **Access to Shape Feather properties via scripting**
  - Added: *Shape.featherSegLocs*
  - Added: *Shape.featherRelSegLocs*
  - Added: *Shape.featherRadii*
  - Added: *Shape.featherInterps*
  - Added: *Shape.featherTensions*
  - Added: *Shape.featherTypes*
  - Added: *Shape.featherRelCornerAngles*

2.31 After Effects 10.5 (CS5.5) (April 2011)

- **Added to the *Project object* object:**
  - *Project.framesCountType*
  - *Project.feetFramesFilmType*
  - *Project.framesUseFeetFrames*
  - *Project.footageTimecodeDisplayStartType*
  - *Project.timeDisplayType*
- **Removed from the *Project object* object:**
  - `timecodeDisplayType` attribute
  - `timecodeBaseType` attribute
  - `timecodeNTSCDropFrame` attribute
  - `timecodeFilmType` attribute
  - `TimecodeDisplayType` enum
  - `TimecodeFilmType` enum
  - `TimecodeBaseType` enum

- Added: *CompItem.dropFrame*
- **Added support for Paragraph Box Text:**
  - Added *LayerCollection.addBoxText()*
  - Added *TextDocument.boxText*
  - Added *TextDocument.pointText*
  - Added *TextDocument.boxTextSize*
- Added *LightLayer.lightType*

2.32 After Effects 9.0 (CS4) (September 2008)

- Added: *app.isoLanguage*
- Added: *MarkerValue.duration*
- Added: *OutputModule.includeSourceXMP*
- Added: *Project.xmpPacket*
- **Added the following Property methods and attributes related to the Separate Dimensions feature:**
  - *Property.dimensionsSeparated*
  - *Property.getSeparationFollower()*
  - *Property.isSeparationFollower*
  - *Property.isSeparationLeader*
  - *Property.separationDimension*
  - *Property.separationLeader*
- **Added *TextDocument object* access, including:**
  - Added: *TextDocument.applyFill*
  - Added: *TextDocument.applyStroke*
  - Added: *TextDocument.fillColor*
  - Added: *TextDocument.font*
  - Added: *TextDocument.fontSize*
  - Added: *TextDocument.justification*
  - Added: *TextDocument.resetCharStyle()*
  - Added: *TextDocument.resetParagraphStyle()*
  - Added: *TextDocument.strokeColor*
  - Added: *TextDocument.strokeOverFill*
  - Added: *TextDocument.strokeWidth*

CHAPTER

THREE

ELEMENTS OF BASIC JAVASCRIPT RELEVANT TO AFTER EFFECTS SCRIPTING

3.1 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.

3.1.1 Keywords and Statement Syntax

Keyword/Statement	Description
`break`	Standard JavaScript; exit the currently executing loop.
`continue`	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.
`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 legalvalue.
`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…whileloop, 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.

3.2 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.

3.2.1 Description of Operators

Operators	Description
`new`	Create new object instance.
`delete`	Delete property from an object.
`typeof`	Returns data type.
`void`	Returns undefined value.
`.`	Object member.
`[]`	Array element.
`()`	Function call.
`++`	Pre- or post-increment.
`--`	Pre- or post-decrement.
`-`	Unary negation or subtraction.
`~`	Bitwise NOT.
`!`	Logical NOT.
`*`	Multiply.
`/`	Divide.
`%`	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.

continues on next page

Table 1 – continued from previous page

Operators	Description
>>=	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.

3.2.2 Operator Precedence

Operators (highest precedence to lowest)	Associativity
[], (), .	left to right
new, delete, - (unary negation), !, type of, 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

3.3 Javascript Classes

3.3.1 Class Inheritance

This is section gives you a brief overview in oriented programming and inheritance. If you already know about this, you can skip this section.

In Javascript/Extendscript, Class Inheritance is the idea that you can define some properties or methods for a given object, and then create a *subclass* (or "child class") that inherits all of those properties & methods and adds more, further refining it.

For example, "automobile" could be one base class, with "cars" being a subclass of the "automobile" base class, with "sedan" and "convertible" being two subclasses of the "car" base class. Any properties or methods from "automobile" are also accessible by "convertible," because there's a direct inheritance from "automobile" -> "car" -> "convertible".

3.3.2 Class Inheritance in After Effects

As a script developer, this is useful to know because many elements in the After Effects scripting environment follows this pattern.

As a user, you can see this in After Effects layers; every layer exists in the timeline, has a Name and Index and Label Color, but some types of layers have different properties than others – for example, Audio layers can't be enable/disabled, and Camera and Light layers can't have effects. They share the base "Layer" features, but are each **subclasses** with their own properties.

The same idea exists in After Effects scripting. Many API-accessible elements are part of class hierarchies that inherit and refine properties & methods. This lets the After Effects developers use existing structures to create new API-accessible components, and it allows script developers to use this same hierarchy to work with the After Effects DOM.

For the same example above, *Layer object* (itself a subclass of *PropertyGroup object*) is the *base class* for *AVLayer object*, *CameraLayer object*, and *LightLayer object*. This means that CameraLayer inherits everything from the Layer object, which inherits everything from the PropertyGroup object, which inherits everything from the PropertyBase object.

This is why you won't see the `name` property on the Layer page, but you can still use `layer.name` in your script; `name` is inherited from *PropertyBase.name*.

> **Warning:** In a few specific cases, properties & methods are **removed** with inheritance, not just added. Those cases are noted on the relevant object page.

3.3.3 Checking Classes

Typically in Javascript, you can use `instanceof` to check whether any given element matches an expected object type.

Keep in mind that you will need to check against the *most specific* class possible; an AE Text Layer will only return `true` for `layer instanceof TextLayer`, and `false` for all parent classes (`layer instanceof AVLayer`, `layer instanceof Layer`, etc.)

With that said, there exist some elements in the API that are *only* base classes for other classes; they exist to hold inherited properties & methods, but no DOM element is exactly of this type.

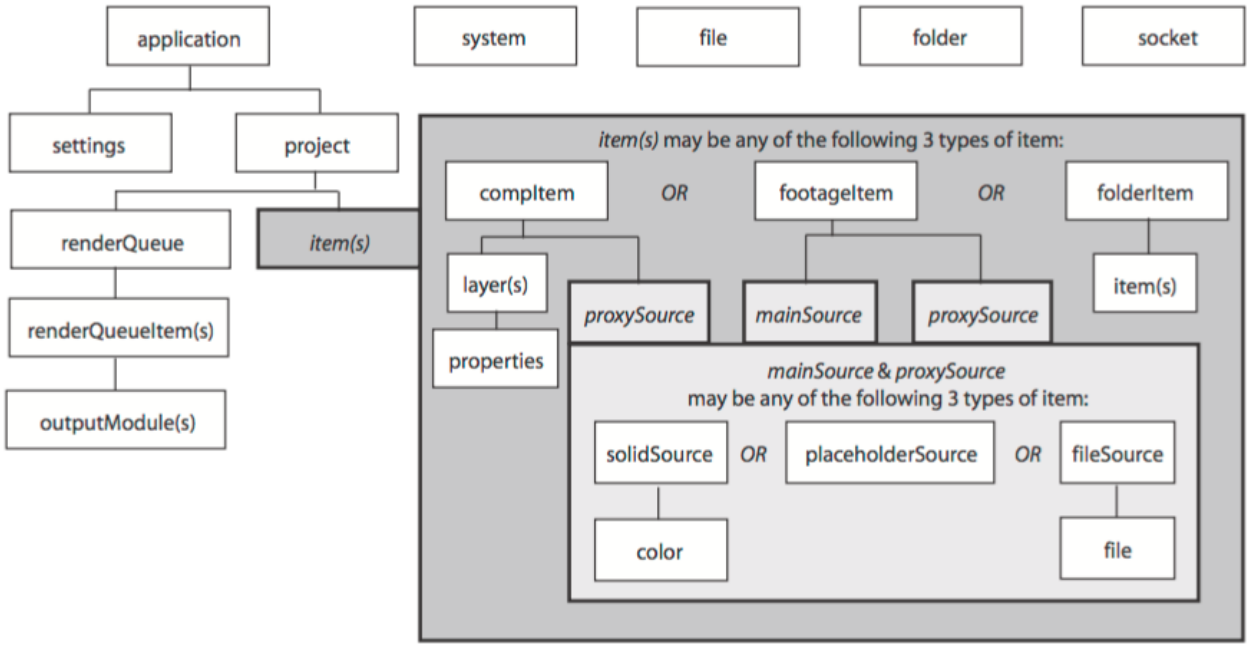
When checking `object instanceof {class}` with these classes, AE will either throw an error that `{class} is undefined` or return `false`, depending on how the class was implemented. The list below documents which base-only classes report which behaviours.

CHAPTER

FOUR

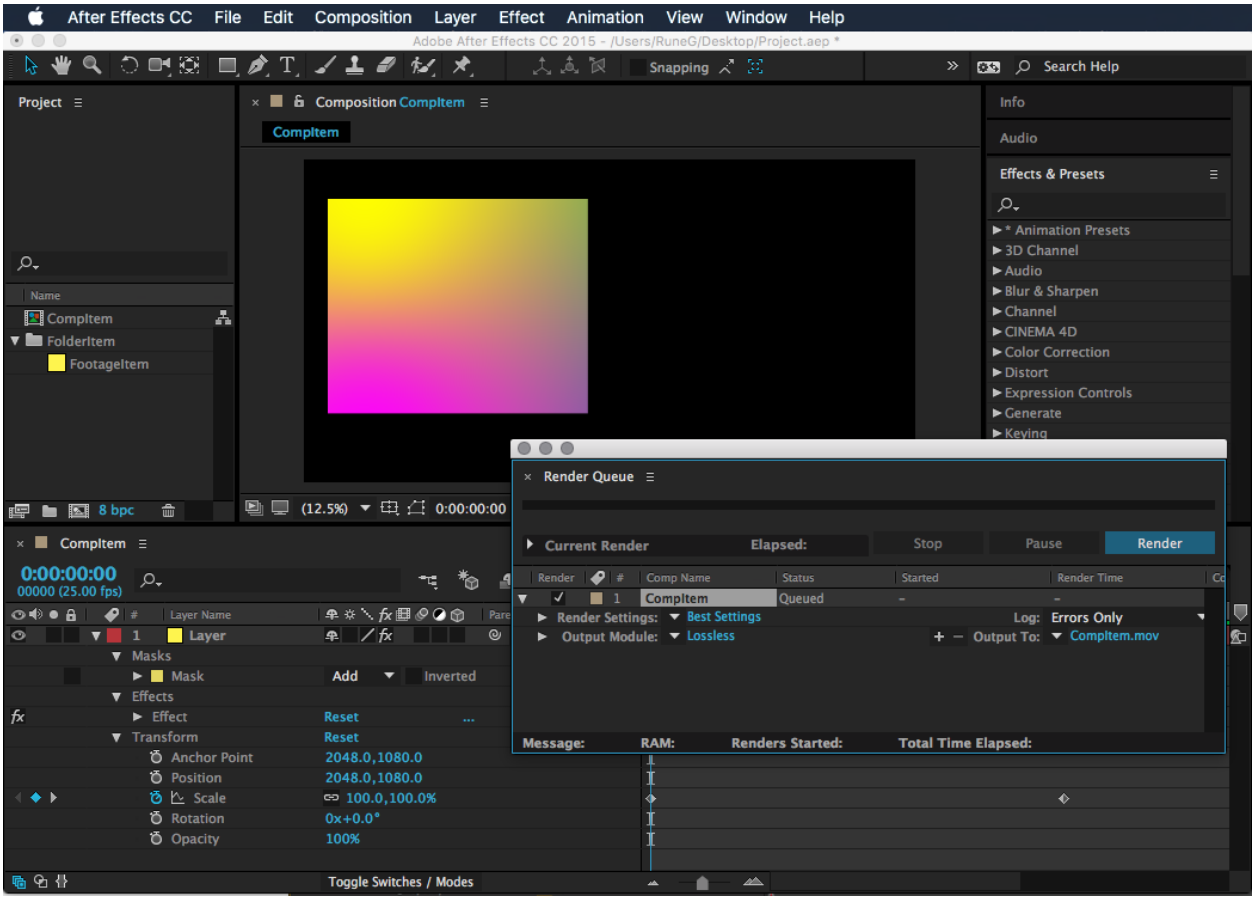
THE AFTER EFFECTS OBJECT MODEL

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 renderqueue 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	Globally available functions that allow you to display text for script debugging purposes, and help conv
Application object	A single global object, available by its name (app), that provides access to objects and application settin
AVItem object	Represents audio/visual files imported into After Effects.
AVLayer object	Represents those layers that contain AVItem objects (composition layers, footage layers, solid layers, tex
CameraLayer object	Represents a camera layer within a composition.
Collection object	Associates a set of objects or values as a logical group and provides access to them by index.
CompItem object	Represents a composition, and allows you to manipulate it and get information about it.
FileSource object	Describes footage that comes from a file.
FolderItem object	Represents a folder in the Project panel.
FootageItem object	Represents a footage item imported into a project, which appears in the Project panel.

Table 1 – continued from previous page

Object	Description
FootageSource object	Describes the file source of some footage.
ImportOptions object	Encapsulates options for importing files into After Effects.
Item object	Represents an item in a project that appears in the Project panel.
ItemCollection object	Collects items in a project.
KeyframeEase object	Encapsulates keyframe ease values in an After Effects property.
Layer object	A base class for layer classes.
LayerCollection object	Collects layers in a project.
LightLayer object	Represents a light layer within a composition.
MarkerValue object	Encapsulates marker values in an After Effects property.
MaskPropertyGroup object	Encapsulates mask attributes in a layer.
OMCollection object	Collects output modules in a render queue.
OutputModule object	Represents an output module for a render queue.
PlaceholderSource object	Describes a placeholder for footage.
Project object	Represents an After Effects project.
Property object	Represents an After Effects property.
PropertyBase object	A base class for After Effects property and property group classes.
PropertyGroup object	Represents an After Effects property group.
RenderQueue object	Represents the After Effects render queue.
RenderQueueItem object	Represents a renderable item in a render queue.
RenderQueueItem object	Collects render-queue items in a render queue.
RQItemCollection object	Provides access to application settings and preferences.
Shape object	Encapsulates the outline shape information for a mask.
ShapeLayer object	Represents a shape layer within a composition.
SolidSource object	Describes a solid color that is the source of some footage.
System object	Provides access to the operating system from the application.
TextDocument object	Encapsulates the text in a text layer.
TextLayer object	Represents a text layer within a composition.
Viewer object	Represents a Composition, Layer, or Footage panel.

CHAPTER

FIVE

AFTER EFFECTS CLASS HIERARCHY

This section lists the class hierarchies for relevant AE API elements. For a primer on what this means, see *Javascript Classes*

When using this guide, any objects that exist as part of a class hierarchy will note whether they exist as a subclass or base class (or both) of another object.

As it can be useful to see all available class hierarchies in one place, we've created this list below.

Note that some classes exist only as base classes, and demonstrate unexpected behaviour when type checking via `instanceof`, as noted in the table below. Classes with no symbol behave as expected.

Symbol Legend

	`instanceof` is always `false`
	Class is undefined; `instanceof` will throw an error

5.1 Properties, Property Groups, and Layers

- *PropertyBase object*
  - *Property object*
  - *PropertyGroup object*
    - *MaskPropertyGroup object*
    - *Layer object*
      - *AVLayer object*
      - *ShapeLayer object*
      - *TextLayer object*
      - *CameraLayer object*
      - *LightLayer object*

5.2 Project Items

- *Item object*
  - *AVItem object*
    - *CompItem object*
    - *FootageItem object*
  - *FolderItem object*

5.3 Footage Item Sources

- *FootageSource object*
  - *FileSource object*
  - *PlaceholderSource object*
  - *SolidSource object*

5.4 Collections

- *Collection object*
  - *ItemCollection object*
  - *LayerCollection object*
  - *OMCollection object*
  - *RQItemCollection object*

CHAPTER

SIX

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, to convert numeric time values to and from string values, or to generate a random number.

Global function	Description
`clearOutput()`	Clears text from the Info panel.
`currentFormatToTime()`	Converts string time value to a numeric time value.
`generateRandomNumber()`	Generates a random number.
`getEnumAsString()`	Converts an Enum value to it's string name.
`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.

6.1 clearOutput()

`clearOutput()`

Description

Clears the output in the Info panel.

Parameters

None.

Returns

Nothing.

6.2 currentFormatToTime()

`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*).

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.

6.3 generateRandomNumber()

`generateRandomNumber()`

Note: This functionality was added in After Effects 13.6 (CC 2015)

Description

Generates random numbers. This function is recommended instead of `Math.random` for generating random numbers that will be applied as values in a project (e.g., when using setValue).

This method avoids a problem where `Math.random` would not return random values in After Effects CC 2015 (13.5.x) due to a concurrency issue with multiple CPU threads.

Returns

Floating-point, pseudo-random number in the range [0, 1].

Example

```
// change the position X of all layers with random number

var myComp = app.project.activeItem;
var x = 0;

for (var i = 1; i <= myComp.numLayers; i++) {
  // If you use Math.random(), this does not work
  // x = 400 * (Math.random()) - 200;
```

(continues on next page)

(continued from previous page)

```
  // use new generateRandomNumber() instead

  x = 400 * generateRandomNumber() - 200;
  var currentPos = myComp.layer(i).property("Position").value;
  myComp.layer(i).property("Position").setValue([currentPos[0] + x, currentPos[1]]);
}
```

6.4 getEnumAsString()

`getEnumAsString()`

Note: This functionality was added in After Effects 24.0.

Description

Returns the string value of an Enum.

Parameters

Enum.

Returns

String.

Example

```
// Returns: "BlendingMode.ADD"
alert(getEnumAsString(5220));
```

6.5 isValid()

`isValid(obj)`

Description

Determines if the specified After Effects object (e.g., composition, layer, mask, etc.) still exists. Some operations, such as *PropertyBase.moveTo()*, 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
```

6.6 timeToCurrentFormat()

`timeToCurrentFormat(time, fps[, isDuration])`

Description

Converts a numeric time value (a number of seconds) to a frame time value; that is, a formatted string thatshows which frame corresponds to that time, at the specified rate. For example, if the time is 0.5 seconds, andthe frame rate is 24 fps, the frame would be 0:00:12 (when the project is set to display as timecode). If the framerate 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).

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.

6.7 write()

`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.");
```

6.8 writeLn()

`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");
```

CHAPTER

SEVEN

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.

7.1 Attributes

7.1.1 app.activeViewer

`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 object; read-only.

7.1.2 app.availableGPUAccelTypes

`app.availableGPUAccelTypes`

Note: This functionality was added in After Effects 14.0 (CC 2017)

Description

Use this in conjunction with `app.project.gpuAccelType` to set the value for Project Settings > Video Rendering and Effects > Use.

Type

Array of `GpuAccelType` enums, or null if no viewers are open; read-only. One of:

- `CUDA`
- `Metal`
- `OPENCL`
- `SOFTWARE`

Example The following sample code checks the current computer's available GPU acceleration types, and sets it to Metal if available.

```
// app.availableGPUAccelTypes returns GPU acceleration types available on the current␣
↪system.
// You can use this to check before setting the GPU acceleration type.
var newType = GpuAccelType.METAL;

// Before trying to set, check which GPU acceleration types are available on the current␣
↪system.
var canSet = false;
var currentOptions = app.availableGPUAccelTypes;
for (var op in currentOptions) {
    if (currentOptions[op] === newType) {
        canSet = true;
    }
}

if (canSet) {
    // Set the GPU acceleration type.
    app.project.gpuAccelType = newType;
} else {
    alert("Metal is not available on this OS.");
}
```

7.1.3 app.buildName

`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.

7.1.4 app.buildNumber

`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.

7.1.5 app.disableRendering

`app.disableRendering`

Note: This functionality was added in After Effects 16.0 (CC 2019)

Description

When false (the default), rendering proceeds as normal. Set to true to disable rendering as if Caps Lock were turned on.

Type

Boolean; read/write.

7.1.6 app.effects

`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.
`version`	Effect's internal version string. This value might be different than the version number the plug-in vendor decides to show in the effect's about box.

Example

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

7.1.7 app.exitAfterLaunchAndEval

`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.

7.1.8 app.exitCode

`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
```

7.1.9 app.fonts

`app.fonts`

Note: This functionality was added in After Effects 24.0.

Description

Returns an object to navigate and retreive all the fonts currently available on your system.

Type

Fonts object; read-only.

7.1.10 app.isoLanguage

`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 running not in English or French.");
}
```

7.1.11 app.isRenderEngine

`app.isRenderEngine`

Description

True if After Effects is running as a render engine.

Type

Boolean; read-only.

7.1.12 app.isWatchFolder

`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.

7.1.13 app.memoryInUse

`app.memoryInUse`

Description

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

Type

Number; read-only.

7.1.14 app.onError

`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()*. 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;
```

7.1.15 app.preferences

`app.preferences`

Description

The currently loaded AE app preferences. See *Preferences object*.

Type

Preferences object; read-only.

7.1.16 app.project

`app.project`

Description

The project that is currently loaded. See *Project object*.

Type

Project object; read-only.

7.1.17 app.saveProjectOnCrash

`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.

7.1.18 app.settings

`app.settings`

Description

The currently loaded settings. See *Settings object*.

Type

Settings object; read-only.

7.1.19 app.version

`app.version`

Note: This functionality was added in After Effects 12.0 (CC)

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 AfterEffects.");
```

7.2 Methods

7.2.1 app.activate()

`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.

7.2.2 app.beginSuppressDialogs()

`app.beginSuppressDialogs()`

Description

Begins suppression of script error dialog boxes in the user interface. Use *app.endSuppressDialogs()* to resume the display of error dialogs.

Parameters

None.

Returns

Nothing.

7.2.3 app.beginUndoGroup()

`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 *app.endUndoGroup()* method to mark the end of the group.

`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 ")

Returns

Nothing.

7.2.4 app.cancelTask()

`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.

7.2.5 app.endSuppressDialogs()

`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 *app.beginSuppressDialogs()* has previously been called.

Parameters

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

Returns

Nothing.

7.2.6 app.endUndoGroup()

`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.

7.2.7 app.endWatchFolder()

`app.endWatchFolder()`

Description

Ends Watch Folder mode.

Parameters

None.

Returns

Nothing.

See also

- *app.watchFolder()*
- *app.parseSwatchFile()*
- *app.isWatchFolder*

7.2.8 app.executeCommand()

`app.executeCommand(id)`

Description

Menu Commands in the GUI application have an individual ID number, which can be used as the parameter for this method. For some functions not included in the API this is the only way to access them.

The *app.findMenuCommandId()* method can be used to find the ID number for a command.

These web sites have more information and lists of the known numbers:

- https://www.provideocoalition.com/after-effects-menu-command-ids/
- https://hyperbrew.co/blog/after-effects-command-ids/

Parameters

id	The ID number of the command.

Returns

None.

Example

```
// calls the Convert to Bezier Path command
app.executeCommand(4162);
```

7.2.9 app.findMenuCommandId()

`app.findMenuCommandId(Command)`

Description

Menu Commands in the GUI application have an individual ID number, which can be used as a parameter for the *app.executeCommand()* command. For some functions not included in the API this is the only way to access them.

It should be noted that this method is not reliable across different language packages of AE, so you'll likely want to find the command ID number during development and then call it directly using the number in production.

These web sites have more information and lists of the known numbers:

- https://www.provideocoalition.com/after-effects-menu-command-ids/
- https://hyperbrew.co/blog/after-effects-command-ids/

Parameters

Command	The text of the menu command, exactly as it is shown in the UI.

Returns

Integer, the ID number of the menu command.

Example

```
app.findMenuCommandId("Convert To Bezier Path")
```

7.2.10 app.newProject()

`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()*

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();
```

7.2.11 app.open()

```
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) {
  var new_project = app.open(my_file);
  if (new_project) {
    alert(new_project.file.name);
  }
}
```

7.2.12 app.parseSwatchFile()

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

7.2.13 app.pauseWatchFolder()

`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

- *app.isWatchFolder*
- *app.watchFolder()*
- *app.endWatchFolder()*

7.2.14 app.purge()

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

Note:

This functionality was updated in After Effects 24.3 to allow the `ALL_CACHES` enumerated value to clear both the RAM and disk cache, with the ALL_MEMORY_CACHES enumerated value added to purge only the RAM.

In versions prior to 24.3, `ALL_CACHES` will only clear the RAM cache.

Description

Purges unused data of the specified types. 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 both RAM and disk cache. `PurgeTarget.ALL_MEMORY_CACHES`: Purges all data that After Effects has cached to RAM. *(new in 24.3)* `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.

7.2.15 app.quit()

```
app.quit()
```

Description

Quits the After Effects application.

Parameters

None.

Returns

Nothing.

7.2.16 app.scheduleTask()

```
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()*.

7.2.17 app.setMemoryUsageLimits()

```
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.

7.2.18 app.setMultiFrameRenderingConfig()

```
app.setMultiFrameRenderingConfig(mfr_on, max_cpu_perc)
```

Note: This functionality was added in After Effects 22.0 (2022)

Description

Calling this function from a script will set the Multi-Frame Rendering configuration for the next render. After execution of the script is complete, these settings will be reset to what was previously set in the UI.

Parameters

mfr_on	Boolean value. Set to true to enable Multi-Frame Rendering.
max_cpu_per	Value from 1-100 representing the maximum CPU percentage Multi-Frame Rendering should utilize. If mfr_on is set to false, pass in 100.

Returns

Nothing.

7.2.19 app.setSavePreferencesOnQuit()

`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.

7.2.20 app.watchFolder()

`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 Folder object for the folder to watch.

Returns

Nothing.

Example

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

See also

- *app.endWatchFolder()*
- *app.parseSwatchFile()*
- *app.isWatchFolder*

CHAPTER

EIGHT

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.

8.1 Attributes

8.1.1 Project.activeItem

`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.

8.1.2 Project.bitsPerChannel

`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.

8.1.3 Project.compensateForSceneReferredProfiles

`app.project.compensateForSceneReferredProfiles`

Note: This functionality was added in After Effects 16.0 (CC 2019)

Description

True if Compensate for Scene-referred Profiles should be enabled for this project; otherwise false.

Type

Boolean; read/write.

8.1.4 Project.dirty

`app.project.dirty`

Note: This functionality was added in After Effects 17.5 (CC2020).

Warning: This method/property is officially undocumented and was found via research. The information here may be inaccurate, and this whole method/property may disappear or stop working some point. Please contribute if you have more information on it!

Description

True if the project has been modified from the last save; otherwise false.

"Dirty" projects will have an * in the project window title.

Type

Boolean; read-only.

8.1.5 Project.displayStartFrame

`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*.

Type

Integer (0 or 1); read/write.

8.1.6 Project.expressionEngine

`app.project.expressionEngine`

Note: This functionality was added in After Effects 16.0 (CC 2019)

Description

The Expressions Engine setting in the Project Settings dialog box, as a string. One of:

- `extendscript`
- `javascript-1.0`

Type

String; read/write.

8.1.7 Project.feetFramesFilmType

`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`

8.1.8 Project.file

`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.

8.1.9 Project.footageTimecodeDisplayStartType

`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`

8.1.10 Project.framesCountType

`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`

> **Warning:** Setting this attribute to `FramesCountType.FC_TIMECODE_CONVERSION` resets the `displayStartFrame` attribute to 0.

8.1.11 Project.framesUseFeetFrames

`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.

8.1.12 Project.gpuAccelType

`app.project.gpuAccelType`

Note: This functionality was added in After Effects 13.8 (CC 2015.3)

Description

Get or set the current projects GPU Acceleration option. see *app.availableGPUAccelTypes*

Type

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

- `GpuAccelType.CUDA`
- `GpuAccelType.Metal`
- `GpuAccelType.OPENCL`
- `GpuAccelType.SOFTWARE`

Example

```
// access via scripting to Project Settings -> Video Rendering and Effects -> Use

var currentGPUSettings = app.project.gpuAccelType; // returns the current value
var type_str = "";

// check the current value and alert the user

switch (currentGPUSettings) {
    case GpuAccelType.CUDA:
        type_str = "CUDA";
        break;
    case GpuAccelType.METAL:
        type_str = "Metal";
        break;
    case GpuAccelType.OPENCL:
        type_str = "OpenCL";
        break;
    case GpuAccelType.SOFTWARE:
        type_str = "Software";
        break;
    default:
        type_str = "UNKNOWN";
}

alert("Your current setting is " + type_str);

// set the value to Metal
app.project.gpuAccelType = GpuAccelType.METAL;
```

8.1.13 Project.items

`app.project.items`

Description

All of the items in the project.

Type

ItemCollection object; read-only.

8.1.14 Project.linearBlending

`app.project.linearBlending`

Description

True if linear blending should be used for this project; otherwise false.

Type

Boolean; read/write.

8.1.15 Project.linearizeWorkingSpace

`app.project.linearizeWorkingSpace`

Note: This functionality was added in After Effects 16.0 (CC 2019)

Description

True if Linearize Working Space should be enabled for this project; otherwise false.

Type

Boolean; read/write.

8.1.16 Project.numItems

`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

```
var numItems = app.project.numItems;
alert("There are " + numItems + " items in this project.")
```

8.1.17 Project.renderQueue

`app.project.renderQueue`

Description

The renderqueue of the project.

Type

RenderQueue object; read-only.

8.1.18 Project.revision

`app.project.revision`

Description

The current revision of the project. Every user action increases the revision number. New project starts at revision 1.

Type

Integer; the current revision version of the project; read-only.

8.1.19 Project.rootFolder

`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.

8.1.20 Project.selection

`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.

8.1.21 Project.timeDisplayType

`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`

8.1.22 Project.toolType

`app.project.toolType`

Note: This functionality was added in After Effects 14.0 (CC 2017)

Description

Get and sets the active tool in the Tools panel.

Type

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

- `ToolType.Tool_Arrow`: Selection Tool
- `ToolType.Tool_Rotate`: Rotation Tool
- `ToolType.Tool_CameraMaya`: Unified Camera Tool
- `ToolType.Tool_CameraOrbit`: Orbit Camera Tool
- `ToolType.Tool_CameraTrackXY`: Track XY Camera Tool
- `ToolType.Tool_CameraTrackZ`: Track Z Camera Tool
- `ToolType.Tool_Paintbrush`: Brush Tool
- `ToolType.Tool_CloneStamp`: Clone Stamp Tool
- `ToolType.Tool_Eraser`: Eraser Tool
- `ToolType.Tool_Hand`: Hand Tool
- `ToolType.Tool_Magnify`: Zoom Tool
- `ToolType.Tool_PanBehind`: Pan Behind (Anchor Point) Tool
- `ToolType.Tool_Rect`: Rectangle Tool
- `ToolType.Tool_RoundedRect`: Rounded Rectangle Tool
- `ToolType.Tool_Oval`: Ellipse Tool
- `ToolType.Tool_Polygon`: Polygon Tool
- `ToolType.Tool_Star`: Star Tool

- `ToolType.Tool_TextH`: Horizontal Type Tool
- `ToolType.Tool_TextV`: Vertical Type Tool
- `ToolType.Tool_Pen`: Pen Tool
- `ToolType.Tool_Feather`: Mask Feather Tool
- `ToolType.Tool_PenPlus`: Add Vertex Tool
- `ToolType.Tool_PenMinus`: Delete Vertex Tool
- `ToolType.Tool_PenConvert`: Convert Vertex Tool
- `ToolType.Tool_Pin`: Puppet Pin Tool
- `ToolType.Tool_PinStarch`: Puppet Starch Tool
- `ToolType.Tool_PinDepth`: Puppet Overlap Tool
- `ToolType.Tool_Quickselect`: Roto Brush Tool
- `ToolType.Tool_Hairbrush`: Refine Edge Tool

Examples

The following sample code checks the current tool, and if it is not the Unified Camera Tool, sets the current tool to that:

```
// Check the current tool, then set it to Unified Camera Tool (UCT).
// Assume a composition is selected in the project.
var comp = app.project.activeItem;
if (comp instanceof CompItem) {
    // Add a camera to the current comp. (Requirement for UCT)
    var cameraLayer = comp.layers.addCamera("Test Camera", [comp.width / 2, comp.height /
↪ 2]);
    comp.openInViewer();

    // If the currently selected tool is not one of the camera tools, set it to UCT.
    if (( app.project.toolType !== ToolType.Tool_CameraMaya) &&
        ( app.project.toolType !== ToolType.Tool_CameraOrbit ) &&
        ( app.project.toolType !== ToolType.Tool_CameraTrackXY) &&
        ( app.project.toolType !== ToolType.Tool_CameraTrackZ)) {
            app.project.toolType = ToolType.Tool_CameraMaya;
        }
}
```

The following sample code uses the new app.project.toolType attribute to create a 360-degrees composition (environment layer and camera) from a selected footage item or composition selected in the Project panel. This script a good starting point for building VR compositions from equirectangular footage:

```
// Create a 360 VR comp from a footage item or comp selected in the Project panel.

var item = app.project.activeItem;
if (item !== null && (item.typeName === "Footage" || item.typeName === "Composition")) {
    // Create a comp with the footage.
    var comp = app.project.items.addComp(item.name, item.width, item.height, item.
↪pixelAspect, item.duration, item.frameRate);
    var layers = comp.layers;
    var footageLayer = layers.add(item);
```

(continues on next page)

(continued from previous page)

```
    // Apply the CC Environment effect and create a camera.
    var effect = footageLayer.Effects.addProperty("CC Environment");
    var camera = layers.addCamera("360 Camera", [item.width / 2, item.height / 2]);
    comp.openInViewer();
    app.project.toolType = ToolType.Tool_CameraMaya;
} else {
    alert("Select a single footage item or composition in the Project panel.");
}
```

8.1.23 Project.transparencyGridThumbnails

`app.project.transparencyGridThumbnails`

Description

When true, thumbnail views use the transparency checkerboard pattern.

Type

Boolean; read/write.

8.1.24 Project.usedFonts

`app.project.usedFonts`

Note: This functionality was added in After Effects (Beta) 24.0 and subject to change while it remains in Beta.

Description

Returns an Array of Objects containing references to used fonts and the Text Layers and times on which they appear in the current *Project*. Each object is composed of `font` which is a *Font object*, and `usedAt` which is an Array of Objects, each composed of `layerID`, a *Layer.id*, and `timeD` for when. See *Project.layerByID()* to retrieve the layers.

```
var usedList = app.project.usedFonts;
if (usedList.length) {
    var font = usedList[0].font;
    var usedAt = usedList[0].usedAt;

    var str = "[0]:" + font.postScriptName + "\n";
    for (var i = 0; i < usedAt.length; i++) {
         var layerID = usedAt[i].layerID;
         var timeD  = usedAt[i].timeD;

         var layer = app.project.layerByID(layerID);
         str += "   Layer:'" + String(layer.property("Source Text").valueAtTime(timeD,␣
↪false)) + "'\n";
    }
    alert(str);
}
```

Type

Array of Objects; read-only.

8.1.25 Project.workingGamma

`app.project.workingGamma`

Description

The current project's working gamma value, either 2.2 or 2.4. Setting values other than 2.2 or 2.4 will cause a scripting error. Note that when the project's color working space is set, the working gamma value is ignored by After Effects.

Type

Number; read/write.

Examples

- To set the working gamma to 2.4 (Rec. 709): `app.project.workingGamma = 2.4;`
- To get the current working gamma: `var currentGamma = app.project.workingGamma;`

8.1.26 Project.workingSpace

`app.project.workingSpace`

Description

A string which is the color profile description for the project's color working space. To set the working space to None, set `workingSpace` to an empty string.

Use `app.project.listColorProfiles()` to return an array of available color profile descriptions that can be used to set the color working space.

Type

String; read/write.

Examples

- To set the working space to Rec.709 Gamma 2.4: `app.project.workingSpace = "Rec.709 Gamma 2.4";`
- To set the working space to None: `app.project.workingSpace = "";`
- To get the current working space: `var currentSpace = app.project.workingSpace;`

8.1.27 Project.xmpPacket

`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 XMPlibrary 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 XMPmetadata
// update the Label project metadata's value
var schemaNS = XMPMeta.getNamespaceURI("xmp");
var propName = "xmp:Label";
try{
    mdata.setProperty(schemaNS, propName, "finalversion...no, really!");
} catch (e) {
    alert(e);
}

app.project.xmpPacket = mdata.serialize();
```

8.2 Methods

8.2.1 Project.autoFixExpressions()

`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.

8.2.2 Project.close()

`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.

8.2.3 Project.consolidateFootage()

`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.

8.2.4 Project.importFile()

`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.

Returns

FootageItem object.

Example

```
app.project.importFile(new ImportOptions(new File("sample.psd"));
```

8.2.5 Project.importFileWithDialog()

`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.

8.2.6 Project.importPlaceholder()

`app.project.importPlaceholder(name, width, height, frameRate, duration)`

Description

Creates and returns a new PlaceholderItem 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.

8.2.7 Project.item()

`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.

8.2.8 Project.itemByID()

`app.project.itemByID(id)`

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Retrieves an item by its *Item ID*

Parameters

`id`	The ID of an item, an integer.

Returns

Item object.

8.2.9 Project.layerByID()

`app.project.layerByID(id)`

Note: This functionality was added in After Effects 22.0 (2022)

Description

Instance method on Project which, when given a valid ID value, returns the Layer object in the Project with that given ID.

Parameters

id	A non-negative integer representing the ID of the Layer to be retrieved from the Project.

Returns

Layer object with the given ID if it exists on the project; otherwise null. Non-valid IDs will throw an exception stating that the input parameter is not an unsigned integer.

Example

```
var firstComp = app.project.item(1);
var firstLayer = firstComp.layer(1);
var layerID = firstLayer.id;

if (app.project.layerByID(layerID) === firstLayer) {
   alert("You can get the Layer from the ID!");
}
```

8.2.10 Project.listColorProfiles()

`app.project.listColorProfiles()`

Description

Returns an array of color profile descriptions that can be set as the project's color working space.

Parameters

None.

Returns

Array of strings.

8.2.11 Project.reduceProject()

`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 items = [];
items[items.length] = app.project.item(1);
items[items.length] = app.project.item(3);
app.project.reduceProject(items);
```

8.2.12 Project.removeUnusedFootage()

`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.

8.2.13 Project.replaceFont()

`app.project.replaceFont(fromFont, toFont, [noFontLocking = false])`

Note: This functionality was added in After Effects (Beta) 24.0 and subject to change while it remains in Beta.

Description

This function will replace all the usages of *Font object* `fromFont` with *Font object* `toFont`.

This operation exposes the same mechanism and policy used for automatic font replacement of missing or substituted fonts and is therefore a complete and precise replacement, even on *TextDocuments* which have mixed styling, preserving the character range the `fromFont` was applied to.

This operation is not undoable.

The optional parameter `noFontLocking` controls what happens when the `toFont` has no glyphs for the text it is applied to. By default a fallback font will be selected which will have the necessary glyphs, but if this parameter is set to true then this fallback will not take place and missing glyphs will result. There is no way at the current time to detect or report this.

Note that when `fromFont` is a substituted font and the `toFont` has the same font properties no fallback can occur and the parameter is ignored and treated as true.

```
var fromFont = app.project.usedFonts[0].font;
var fontList = app.fonts.getFontsByPostScriptName("TimesNewRomanPSMT");
var toFont = fontList[0];
var layerChanged = app.project.replaceFont(fromFont, toFont);
```

Parameters

`fromFont`	A *Font object* to be replaced.
`toFont`	A *Font object* to replace it with.
`noFontLocking`	An optional Boolean, defaults to false

Returns

Boolean. True if at least one Layer was changed.

8.2.14 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.

8.2.15 Project.saveWithDialog()

`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.

8.2.16 Project.setDefaultImportFolder()

`app.project.setDefaultImportFolder(folder)`

Description

Sets the folder that will be shown in the file import dialog. This location will be used as an override until setDefaultImportFolder() is called with no parameters, or until After Effects is quit.

Parameters

`folder`	Extendscript Folder object.

Returns

Boolean; indicates if the operation was successful.

Examples

Any of the following will set the default import folder to C:/My Folder:

- `var myFolder = new Folder("C:/My Folder"); app.project.setDefaultImportFolder(myFolder);`
- `app.project.setDefaultImportFolder(new Folder("C:/My Folder"));`
- `app.project.setDefaultImportFolder(Folder("C:/My Folder"));`

Note: if the path refers to an existing file and not a folder, the Folder function returns a File object instead of a Folder object, which will cause `setDefaultImportFolder()` to return false.

To set the default import folder to the current user's desktop folder: `app.project.setDefaultImportFolder(Folder.desktop);`

To disable the default folder, call `setDefaultImportFolder()` with no parameters: `app.project.setDefaultImportFolder();`

8.2.17 Project.showWindow()

`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.

8.3 Team Projects

8.3.1 Project.newTeamProject()

```
app.project.newTeamProject(teamProjectName, description)
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Creates a new team project.

Parameters

`teamProjectName`	Team project name, string value.
`description`	Optional. Team project description, string value.

Returns

Boolean. `True` if the team project is successfully created, `false` otherwise.

8.3.2 Project.openTeamProject()

```
app.project.openTeamProject(teamProjectName)
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Opens a team project.

Parameters

`teamProjectName`	Team project name, string value.

Returns

Boolean. `True` if the team project is successfully opened, `false` otherwise.

8.3.3 Project.shareTeamProject()

```
app.project.shareTeamProject(comment)
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Shares the currently open team project.

Parameters

`comment`	Comment, string value. Optional.

Returns

Boolean. `True` if the team project is successfully shared, `false` otherwise.

8.3.4 Project.syncTeamProject()

```
app.project.syncTeamProject()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Syncs the currently open team project.

Returns

Boolean. `True` if the team project is successfully synced, `false` otherwise.

8.3.5 Project.closeTeamProject()

```
app.project.closeTeamProject()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Closes a currently open team project.

Returns

Boolean. `True` if the team project is successfully closed, `false` otherwise.

8.3.6 Project.convertTeamProjectToProject()

```
app.project.convertTeamProjectToProject(project_file)
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Converts a team project to an After Effects project on a local disk.

Parameters

`project_file`	File object for the local After Effects project. File extension should be either .aep or .aet (.aepx is not supported).

Returns

Boolean. `True` if the team project is successfully converted, `false` otherwise.

8.3.7 Project.listTeamProjects()

```
app.project.listTeamProjects()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Returns an array containing the name strings for all team projects available for the current user. Archived Team Projects are not included.

Returns

Array of strings.

8.3.8 Project.isTeamProjectOpen()

```
app.project.isTeamProjectOpen(teamProjectName)
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Checks whether specified team project is currently open.

Parameters

`teamProjectName`	Team project name, string value.

Returns

Boolean. `True` if the specified team project is currently open, `false` otherwise.

8.3.9 Project.isAnyTeamProjectOpen()

```
app.project.isAnyTeamProjectOpen()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Checks whether any team project is currently open.

Returns

Boolean. `True` if any team project is currently open, `false` otherwise.

8.3.10 Project.isTeamProjectEnabled()

```
app.project.isTeamProjectEnabled()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Checks whether or not team project is enabled for After Effects. (This will almost always return true.)

Returns

Boolean. `True` if team project is currently enabled, `false` otherwise.

8.3.11 Project.isLoggedInToTeamProject()

```
app.project.isLoggedInToTeamProject()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Checks whether or not the client (After Effects) is currently logged into the team project server.

Returns

Boolean. `True` if the client (After Effects) is currently logged into the team projects server, `false` otherwise.

8.3.12 Project.isSyncCommandEnabled()

```
app.project.isSyncCommandEnabled()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Checks whether or not the Sync command is enabled.

Returns

Boolean. `True` if the team projects Sync command is enabled, `false` otherwise.

8.3.13 Project.isShareCommandEnabled()

```
app.project.isShareCommandEnabled()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Checks whether or not the Share command is enabled.

Returns

Boolean. `True` if the team projects Share command is enabled, `false` otherwise.

8.3.14 Project.isResolveCommandEnabled()

```
app.project.isResolveCommandEnabled()
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Checks whether or not the Resolve command is enabled.

Returns

Boolean. `True` if the team projects Resolve command is enabled, `false` otherwise.

8.3.15 Project.resolveConflict()

```
app.project.resolveConflict(ResolveType)
```

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

Resolves a conflict between the open team project and the version on the team projects server, using the specified resolution method.

Parameters

`ResolveType`	The type of conflict resolution to use. A `ResolveType` enumerated value, one of: • `ResolveType.ACCEPT_THEIRS`: Take the shared version. The shared version replaces your version. • `ResolveType.ACCEPT_YOURS`: Keep your version of the project. The shared version is not taken. • `ResolveType.ACCEPT_THEIRS_AND_COPY`: Copy and rename your version, then take the shared version. The shared version replaces your original version

Returns

Boolean. `True` if the resolution of the specified type was successful, `false` otherwise.

CHAPTER

NINE

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 + ".");
```

9.1 Attributes

9.1.1 System.machineName

`system.machineName`

Description

The name of the computer on which After Effects is running.

Type

String; read-only.

9.1.2 System.osName

`system.osName`

Description

The name of the operating system on which After Effects is running.

> **Warning:** As of Windows 7, this attribute returns a blank value. Use $.os instead.

Type

String; read-only.

9.1.3 System.osVersion

`system.osVersion`

Description

The version of the current local operating system.

Type

String; read-only.

9.1.4 System.userName

`system.userName`

Description

The name of the user currently logged on to the system.

Type

String; read-only.

9.2 Methods

9.2.1 System.callSystem()

`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.

CHAPTER

TEN

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 object* and for *FolderItem object*, 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.

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 instanceof FolderItem)) {
    alert("error: second item is not a folder");
} else {
    var numInFolder = myFolder.numItems;
    //Always run loops backwards when deleting things:
    for (var 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) {
                //found an unselected solid.
                curItem.remove();
            }
        }
    }
}
```

10.1 Attributes

10.1.1 Item.comment

`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.

10.1.2 Item.dynamicLinkGUID

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

Description

A unique and persistent identification number used for the dynamic link, in form of `00000000-0000-0000-0000-000000000000`.

Type

String; read-only.

10.1.3 Item.guides

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

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

An array of `guide` objects, containing `orientationType`, `positionType`, and `position` attributes.

Type

Array; read-only.

10.1.4 Item.id

`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.

10.1.5 Item.label

`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).

Note: Custom label colors cannot be set programmatically.

Type

Integer (0 to 16); read/write.

10.1.6 Item.name

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

Description

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

Type

String; read/write.

10.1.7 Item.parentFolder

`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 *ItemCollection.addFolder()* to add a new folder, and set this value to put items in the new folder.

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;
  }
}
```

10.1.8 Item.selected

`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.

10.1.9 Item.typeName

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

Description

A user-readable name for the item type; for example, “Folder”, “Footage”, or “Composition”. These names are application locale-dependent, meaning that they are different depending on the application’s interface language.

Type

String; read-only.

Localized strings

en_US	**Composition**	**Folder**	**Footage**
de_DE	Komposition	Ordner	Footage
es_ES	Composición	Carpeta	Material de archivo
fr_FR	Composition	Dossier	Métrage
it_IT	Composizione	Cartella	Metraggio
ja_JP			
ko_KR			
pt_BR	Composição	Pasta	Gravação
ru_ru			
zh_CN			

Example

```
if (/Composition|Komposition|Composición|Composizione|||Composição||/.test(app.project.
↪item(index).typeName)) {
    // item is a composition
} else if (/Folder|Ordner|Carpeta|Dossier|Cartella|||Pasta||/.test(app.project.
↪item(index).typeName)) {
    // item is a folder
}
```

10.2 Methods

10.2.1 Item.addGuide()

`app.project.item(index).addGuide(orientationType, position)`

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

Creates a new guide and adds it to the `guides` object of the Item.

Parameters

`orientationType`	An integer; 0 for a horizontal guide, 1 for a vertical guide. Any other value defaults to horizontal.
`position`	An integer; the X or Y coordinate position of the guide in pixels, depending on its `orientationType`.

Returns

Integer; the index of the newly-created guide.

Example

Adds a vertical guide at 500 pixels on the X axis to the `activeItem` of a project.

```
app.project.activeItem.addGuide(1, 500);
```

10.2.2 Item.remove()

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

Description

Deletes this item from the project and 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 the disk.

Parameters

None.

Returns

Nothing.

10.2.3 Item.removeGuide()

`app.project.item(index).removeGuide(guideIndex)`

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

Removes an existing guide. Choose the guide based on its index inside the `Item.guides` object.

Parameters

`guideIndex`	An integer; the index of the guide to be removed.

Returns

Nothing.

Example

Removes the first guide in `activeItem`.

```
app.project.activeItem.removeGuide(0);
```

Warning: Removing a guide will cause all higher guide indexes to shift downward.

10.2.4 Item.setGuide()

`app.project.item(index).setGuide(position,guideIndex)`

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

Modifies the `position` of an existing guide. Choose the guide based on its `guideIndex` inside the `Item.guides` array.

A guide's `orientationType` may not be changed after it is created.

Parameters

`position`	An integer; the new X or Y coordinate position of the guide in pixels, depending on its existing `orientationType`.
`guideIndex`	An integer; the index of the guide to be modified.

Returns

Nothing.

Example

Changes the position of the first guide in `activeItem` to 1200 pixels.

```
app.project.activeItem.setGuide(1200, 0);
```

CHAPTER

ELEVEN

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 object*. All methods and attributes of Collection, in addition to those listed below, are available when working with ItemCollection.

11.1 Methods

11.1.1 ItemCollection.addComp()

`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.

11.1.2 ItemCollection.addFolder()

`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.parentFolder* attribute

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
//comp Item'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;
  }
}
```

CHAPTER

TWELVE

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*
>
> 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* and *FootageItem object*.

Warning: CompItems and FootageItems, while logical descendants of AVItem, are not *really* subclasses of AVItem as AVItem doesn't exist in Extendscript, ie. attempting to check if `item instanceof AVItem` will fail because AVItem is undefined. This is also true for `Item` itself.

See *Javascript Classes* and *After Effects Class Hierarchy* for more info.

12.1 Attributes

12.1.1 AVItem.duration

`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.

12.1.2 AVItem.footageMissing

`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* and *FileSource.missingFootagePath*.

Type

Boolean; read-only.

12.1.3 AVItem.frameDuration

`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.

12.1.4 AVItem.frameRate

`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.

12.1.5 AVItem.hasAudio

`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.

12.1.6 AVItem.hasVideo

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

Description

When true, the AVItem has a 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.

12.1.7 AVItem.height

`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.

12.1.8 AVItem.isMediaReplacementCompatible

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

Note: This functionality was added in After Effects 18.0 (2021)

Description

Test whether the AVItem can be used as an alternate source when calling *Property.setAlternateSource()*.

Returns true if the item can be used, or false otherwise.

A CompItem or a FootageItem can be used as an alternate source for the layer, with some restrictions:

- If the AVItem is a *FootageItem object*, then FootageItem.FootageSource should not be a *SolidSource object*.
- If the AVItem is a *FootageItem object* and the FootageItem.FootageSource is a *FileSource object* then that FileSource should not point to a non-media file e.g. a JSX script file.
- Setting the AVItem cannot create a cyclical reference within the project.

Type

Boolean; read only.

12.1.9 AVItem.name

`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.

12.1.10 AVItem.pixelAspect

`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.

12.1.11 AVItem.proxySource

`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.

12.1.12 AVItem.time

`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. 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.

12.1.13 AVItem.usedIn

`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.

12.1.14 AVItem.useProxy

`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.

12.1.15 AVItem.width

`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.

12.2 Methods

12.2.1 AVItem.setProxy()

`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 `mainSource`, 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.

12.2.2 AVItem.setProxyToNone()

`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.

12.2.3 AVItem.setProxyWithPlaceholder()

`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.

12.2.4 AVItem.setProxyWithSequence()

`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.

12.2.5 AVItem.setProxyWithSolid()

`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.

CHAPTER

THIRTEEN

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 object*, which is a subclass of *Item object*. All methods and attributes of AVItem and Item, in addition to those listed below, are available when working with CompItem.

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);
```

13.1 Attributes

13.1.1 CompItem.activeCamera

```
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.

13.1.2 CompItem.bgColor

`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.

13.1.3 CompItem.counters

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

Note: This functionality was added in After Effects 13.2 (CC2014).

Warning: This method/property is officially undocumented and was found via research. The information here may be inaccurate, and this whole method/property may disappear or stop working some point. Please contribute if you have more information on it!

Description

This attribute works app-wide: if changed on one CompItem, it will change it for every CompItem in the project. The value stays until restarting AE. Once restarted, it will revert to false.

This parameter doesn't do anything.

Type

Boolean; read/write.

13.1.4 CompItem.displayStartFrame

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

Description

The frame value of the beginning of the composition.

This value is an alternative to calculating the start frame using *CompItem.displayStartTime* and *CompItem.frameDuration* to compensate for floating-point problems.

Note: This functionality was added in After Effects 17.1.

Type

Integer; read/write.

13.1.5 CompItem.displayStartTime

`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.

Note: As of After Effects 17.1, the minimum value is `-10800.0`. Before 17.1, the minimum value was 0.0

Type

Floating-point value in the range `[-10800.0...86339.0]` (-3:00:00:00 to 23:59:00:00); read/write.

13.1.6 CompItem.draft3d

`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.

13.1.7 CompItem.dropFrame

`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.

13.1.8 CompItem.frameBlending

`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.

13.1.9 CompItem.frameDuration

`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.

13.1.10 CompItem.hideShyLayers

`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.

13.1.11 CompItem.layers

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

Description

A *LayerCollection object* that contains all the Layer objects for layers in this composition.

Type

LayerCollection object; read-only.

13.1.12 CompItem.markerProperty

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

Note: This functionality was added in After Effects 14.0 (CC 2017)

Description

A *PropertyGroup object* that contains all a composition's markers. Composition marker scripting has the same functionality as *Layer markers*.

See *MarkerValue object*.

Type

PropertyGroup object or null; read-only.

Example

The following sample code creates a project and composition, then creates two composition markers with different properties

```
// comp.markerProperty allows you to add markers to a comp.
// It has the same functionality as layer.property("Marker")
var currentProj = app.newProject();
var comp = currentProj.items.addComp("mycomp", 1920, 1080, 1.0, 5, 29.97);
var solidLayer = comp.layers.addSolid([1, 1, 1], "mylayer", 1920, 1080, 1.0);

var compMarker = new MarkerValue("This is a comp marker!");
compMarker.duration = 1;

var compMarker2 = new MarkerValue("Another comp marker!");
compMarker2.duration = 1;

comp.markerProperty.setValueAtTime(1, compMarker);
comp.markerProperty.setValueAtTime(3, compMarker2);
```

13.1.13 CompItem.motionBlur

`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.

13.1.14 CompItem.motionBlurAdaptiveSampleLimit

`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.

13.1.15 CompItem.motionBlurSamplesPerFrame

`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.

13.1.16 CompItem.motionGraphicsTemplateControllerCount

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

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

The number of properties in the Essential Graphics panel for the composition.

Type

Integer; read-only.

13.1.17 CompItem.motionGraphicsTemplateName

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

Note: This functionality was added in After Effects 15.0 (CC 2018)

Description

Read or write the name property in the Essential Graphics panel for the composition.

The name in the Essential Graphics panel is used for the file name of an exported Motion Graphics template (ex., "My Template.mogrt").

The following example will set the name for the active composition and then return it as an alert

```
app.project.activeItem.motionGraphicsTemplateName = "My Template";
alert(app.project.activeItem.motionGraphicsTemplateName);
```

Type

String; read/write.

13.1.18 CompItem.numLayers

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

Description

The number of layers in the composition.

Type

Integer; read-only.

13.1.19 CompItem.preserveNestedFrameRate

`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.

13.1.20 CompItem.preserveNestedResolution

`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.

13.1.21 CompItem.renderer

`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.

13.1.22 CompItem.renderers

`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.

13.1.23 CompItem.resolutionFactor

`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.

13.1.24 CompItem.selectedLayers

`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.

13.1.25 CompItem.selectedProperties

`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.

13.1.26 CompItem.shutterAngle

`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.

13.1.27 CompItem.shutterPhase

`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.

13.1.28 CompItem.workAreaDuration

`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.

13.1.29 CompItem.workAreaStart

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

Description

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

Type

Floating-point; read/write.

13.2 Methods

13.2.1 CompItem.duplicate()

`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.

13.2.2 CompItem.exportAsMotionGraphicsTemplate()

`app.project.item(index).exportAsMotionGraphicsTemplate(doOverWriteFileIfExisting, file_path)`

Note: This functionality was added in After Effects 15.0 (CC 2018)

Description

Exports the composition as a Motion Graphics template. Returns true if the Motion Graphics template is successfully exported, false otherwise.

The name in the Essential Graphics panel is used for the file name of the Motion Graphics template (ex., "My Template.mogrt"). Use the `motionGraphicsTemplateName` attribute to set the name.

Optionally specify the path to the folder where the Motion Graphics template file is saved. If not specified, the file will be saved in the current user's Essential Graphics folder:

```
macOS: /Users/<name>/Library/Application Support/Adobe/Common/Essential Graphics/
Windows: C:\Users\<name>\AppData\Roaming\Adobe\Common\Essential Graphics\
```

If the project has been changed since the last time it was saved, After Effects will prompt the user to save the project. To avoid this, use the project `save()` method before exporting the Motion Graphics template.

Parameters

`doOverWriteFileIfExisting`	Whether to overwrite an exsiting file of the same name, boolean. Required.
`file_path`	Path to the folder where the file will be saved. Optional.

Returns

Boolean.

13.2.3 CompItem.getMotionGraphicsTemplateControllerName()

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

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

Gets the name of a single property in the Essential Graphics panel.

Parameters

`index`	Integer; the index of the EGP property whose name will be returned.

Returns

String; read-only.

13.2.4 CompItem.setMotionGraphicsControllerName()

```
app.project.item(index).setMotionGraphicsControllerName(index,newName)
```

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

Sets the name of a single property in the Essential Graphics panel.

Note: To rename a property as it is added to the EGP, see *Property.addToMotionGraphicsTemplateAs()*.

Parameters

`index`	Integer; the index of the EGP property to be renamed.
`newName`	String; the new name for the EGP property.

Returns

String; read-only.

13.2.5 CompItem.layer()

```
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 thislayer to findthe positionof the desired layer.
`relIndex`	The postion 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.

13.2.6 CompItem.openInEssentialGraphics()

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

Note: This functionality was added in After Effects 15.0 (CC 2018)

Description

Opens the composition in the Essential Graphics panel.

Parameters

None.

Returns

Nothing.

13.2.7 CompItem.openInViewer()

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

CHAPTER

FOURTEEN

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 (var i = 1; i <= secondItem.numItems; i++) {
        alert("item number " + i + " within the folder is named " + secondItem.item(i).
↪name);
    }
}
```

14.1 Attributes

14.1.1 FolderItem.items

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

Description

An ItemCollection object containing Item object that represents 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. The 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.

14.1.2 FolderItem.numItems

`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.

14.2 Methods

14.2.1 FolderItem.item()

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

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

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

Returns Item object.

CHAPTER

FIFTEEN

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 object*, which is a subclass of *Item object*. All methods and attributes of AVItem and Item, in addition to those listed below, are available when working with FootageItem.

15.1 Attributes

15.1.1 FootageItem.file

`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.

15.1.2 FootageItem.mainSource

`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*, and its three types:

- *SolidSource object*
- *FileSource object*
- *PlaceholderSource object*

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.

Type

FootageSource object; read-only.

15.2 Methods

15.2.1 FootageItem.openInViewer()

```
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.

15.2.2 FootageItem.replace()

```
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*) 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.

15.2.3 FootageItem.replaceWithPlaceholder()

`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]`.

15.2.4 FootageItem.replaceWithSequence()

`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*) 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.

15.2.5 FootageItem.replaceWithSolid()

`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].

CHAPTER

SIXTEEN

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 a subclass of *PropertyGroup*, which is a subclass of *PropertyBase*. All methods and attributes of PropertyGroup, in addition to those listed below, are available when working with Layer, with the exception that `propertyIndex` attribute is set to `undefined`.
>
> Layer is the base class for *CameraLayer object*, *LightLayer object*, and *AVLayer object*, so Layer attributes and methods are available when working with all layer types. Layers contain AE properties, in addition to their JavaScript attributes and methods. For examples of how to access properties in layers, see *PropertyBase object*.

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 = "DisabledLayer";
```

16.1 Attributes

16.1.1 Layer.autoOrient

`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.

16.1.2 Layer.comment

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

Description

A descriptive comment for the layer.

Type

String; read/write.

16.1.3 Layer.containingComp

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

Description

The composition that contains this layer.

Type

CompItem object; read-only.

16.1.4 Layer.hasVideo

`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.

16.1.5 Layer.id

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

Note: This functionality was added in After Effects 22.0 (2022)

Description

Instance property on Layer which returns a unique and persistent identification number used internally to identify a Layer 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 Layers in the imported project. The ID is not displayed anywhere in the user interface..

Type

Integer; read-only.

16.1.6 Layer.index

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

Description

The index position of the layer.

Type

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

16.1.7 Layer.inPoint

`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.

16.1.8 Layer.isNameSet

`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.

Note: This always returns *true* for layers that do not have a *AVLayer.source*

Type

Boolean; read-only.

16.1.9 Layer.label

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

Note: Custom label colors cannot be set programmatically.

Type

Integer (0 to 16); read/write.

16.1.10 Layer.locked

`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.

16.1.11 Layer.marker

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

Description

A *PropertyGroup object* that contains all a layer's markers. Layer marker scripting has the same functionality as *Comp markers*.

See *MarkerValue object*.

Type

PropertyGroup object or null; read-only.

Example

The following sample code creates two layer markers with different properties

```
var solidLayer = comp.layers.addSolid([1, 1, 1], "mylayer", 1920, 1080, 1.0);

var layerMarker = new MarkerValue("This is a layer marker!");
layerMarker.duration = 1;

var layerMarker2 = new MarkerValue("Another comp marker!");
layerMarker2.duration = 1;

solidLayer.marker.setValueAtTime(1, layerMarker);
solidLayer.marker.setValueAtTime(3, layerMarker2);
```

16.1.12 Layer.nullLayer

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

Description

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

Type

Boolean; read-only.

16.1.13 Layer.outPoint

`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.

16.1.14 Layer.parent

`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.

16.1.15 Layer.selectedProperties

`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.

16.1.16 Layer.shy

`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.

16.1.17 Layer.solo

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

Description

When true, the layer is soloed, otherwise false.

Type

Boolean; read/write.

16.1.18 Layer.startTime

`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.

16.1.19 Layer.stretch

`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.

16.1.20 Layer.time

`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.

16.2 Methods

16.2.1 Layer.activeAtTime()

`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.

16.2.2 Layer.applyPreset()

```
app.project.item(index).layer(index).applyPreset(presetName);
```

Description

Applies the specified collection of animation settings (an animation preset) to all the currently selected layers of the comp to which the layer belongs. If no layer is selected, it applies the animation preset to a new solid layer.

Predefined animation preset files are installed in the Presets folder, and users can create new animation presets through the user interface.

Warning: The animation preset is applied to the the selected layer(s) of the comp, not to the layer whose applyPreset function is called. Hence, the layer whose applyPreset function is called effectively just determines the comp whose layers are processed.

Parameters

`presetName`	An Extendscript File object for the file containing the animation preset.

Returns

Nothing.

16.2.3 Layer.copyToComp()

```
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 *LayerCollection object* in the target CompItem. Retrieve the copy using into `Comp.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.

Note: As of After Effects 13.6, this method no longer causes After Effects to crash when the layer has a parent.

Warning: As of After Effects 13.7 (13.6, has not been tested), if the copied layer has an effect on it and the user undoes the action, After Effects will Crash.

Tip: The scripting guide says this method copies the layer to the top of the comp. It actually copies it to above the first selected layer, or to the top, if nothing is selected. To retrieve the copy you have to check `CompItem.selectedLayers` for the layer with the topmost index, and use `comp.layer( topmost_index_of_selected_layers - 1 )` to retrieve the layer.

Parameters

intoComp	The target composition, and *CompItem object*.

Returns

Nothing.

16.2.4 Layer.doSceneEditDetection()

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

Note: This functionality was added in After Effects 22.3 (2022)

Description

Runs Scene Edit Detection on the layer that the method is called on and returns an array containing the times of any detected scenes. This is the same as selecting a layer in the Timeline and choosing "Layer > Scene Edit Detection" with the single argument determining whether the edits are applied as markers, layer splits, pre-comps, or are not applied to the layer.

Just as in the UI, `doSceneEditDetection` will fail and error if called on a non-video layer or a video layer with Time Remapping enabled.

Parameters

applyOptions	How the detected edits will be applied. A `SceneEditDetectionMode` enumerated value, one of: • `SceneEditDetectionMode.MARKERS`: Create markers at edit points. • `SceneEditDetectionMode.SPLIT`: Split layer . • `SceneEditDetectionMode.SPLIT_PRECOMP`: Split layer at edit points and pre-compose each one. • `SceneEditDetectionMode.NONE`: Detected edits are not applied to the layer.

Returns

Array of floating-point values; the times of the detected edit points expressed in composition time.

16.2.5 Layer.duplicate()

`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.

16.2.6 Layer.moveAfter()

`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.

16.2.7 Layer.moveBefore()

`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.

16.2.8 Layer.moveToBeginning()

```
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.

16.2.9 Layer.moveToEnd()

```
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.

16.2.10 Layer.remove()

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

Description

Deletes the specified layer from the composition.

Parameters

None.

Returns

Nothing.

16.2.11 Layer.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.

CHAPTER

SEVENTEEN

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 object*. All methods and attributes of Collection, in addition to those listed below, are available when working with LayerCollection.

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);
```

17.1 Methods

17.1.1 LayerCollection.add()

`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 du r at i on 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;

17.1.2 LayerCollection.addBoxText()

`app.project.item(index).layers.addBoxText([width, height])`

Description

Creates a new paragraph (box) text layer with *TextDocument.lineOrientation* set to `LineOrientation.HORIZONTAL` and adds the new *TextLayer object* to this collection. To create a point text layer, use the *LayerCollection.addText()* method.

Parameters

[width, height]	An Array containing the dimensions of the new text box.

Returns

TextLayer object.

17.1.3 LayerCollection.addCamera()

`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.

17.1.4 LayerCollection.addLight()

`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.

17.1.5 LayerCollection.addNull()

`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.

17.1.6 LayerCollection.addShape()

`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.

17.1.7 LayerCollection.addSolid()

`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 Footage Item 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 (MacOS), and specify options under Still Footage.

Returns

AVLayer object.

17.1.8 LayerCollection.addText()

`app.project.item(index).layers.addText([sourceText])`

Description

Creates a new point text layer with *TextDocument.lineOrientation* set to `LineOrientation.HORIZONTAL` and adds the new *TextLayer object* to this collection. To create a paragraph (box) text layer, use *LayerCollection.addBoxText()*.

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.

Returns

TextLayer object.

17.1.9 LayerCollection.addVerticalBoxText()

```
app.project.item(index).layers.addVerticalBoxText([width, height])
```

Note: This functionality was added in After Effects 24.2.

Description

Creates a new paragraph (box) text layer with *TextDocument.lineOrientation* set to `LineOrientation.VERTICAL_RIGHT_TO_LEFT` and adds the new *TextLayer object* to this collection. To create a point text layer, use the *LayerCollection.addText()* or *LayerCollection.addVerticalText()* methods.

Parameters

`[width, height]`	An Array containing the dimensions of the new text box.

Returns

TextLayer object.

17.1.10 LayerCollection.addVerticalText()

```
app.project.item(index).layers.addVerticalText([sourceText])
```

Note: This functionality was added in After Effects 24.2.

Description

Creates a new point text layer with *TextDocument.lineOrientation* set to `LineOrientation.VERTICAL_RIGHT_TO_LEFT` and adds the new *TextLayer object* to this collection. To create a paragraph (box) text layer, use the *LayerCollection.addBoxText()* or *LayerCollection.addVerticalBoxText()* methods.

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.

Returns

TextLayer object.

17.1.11 LayerCollection.byName()

`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.

17.1.12 LayerCollection.precompose()

`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

`layerIndicies`	The composition 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.

CHAPTER

EIGHTEEN

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 object*. All methods and attributes of Layer, in addition to those listed below, are available when working with AVLayer.
>
> AVLayer is a base class for *TextLayer object*, so AVLayer attributes and methods are available when working with TextLayer objects.

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
  - AudioLevels

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;
```

18.1 Attributes

18.1.1 AVLayer.adjustmentLayer

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

Description

True if the layer is an adjustment layer.

Type

Boolean; read/write.

18.1.2 AVLayer.audioActive

`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.

18.1.3 AVLayer.audioEnabled

`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.

18.1.4 AVLayer.blendingMode

`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.DIVIDE
- 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.SUBTRACT
- BlendingMode.SILHOUETE_ALPHA - note the mispelling of 'SILHOUETTE'!
- BlendingMode.SILHOUETTE_LUMA
- BlendingMode.SOFT_LIGHT
- BlendingMode.STENCIL_ALPHA
- BlendingMode.STENCIL_LUMA
- BlendingMode.SUBTRACT
- BlendingMode.VIVID_LIGHT

18.1.5 AVLayer.canSetCollapseTransformation

`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.

18.1.6 AVLayer.canSetTimeRemapEnabled

`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.

18.1.7 AVLayer.collapseTransformation

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

Description

True if collapse transformation is on for this layer.

Type

Boolean; read/write.

18.1.8 AVLayer.effectsActive

`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.

18.1.9 AVLayer.environmentLayer

`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.

18.1.10 AVLayer.frameBlending

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

Description

True if frame blending is enabled for the layer.

Type

Boolean; read-only.

18.1.11 AVLayer.frameBlendingType

`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`

18.1.12 AVLayer.guideLayer

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

Description

True if the layer is a guide layer.

Type

Boolean; read/write.

18.1.13 AVLayer.hasAudio

`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.

18.1.14 AVLayer.hasTrackMatte

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

Note: This functionality was updated in After Effects 23.0. Track Matte is no longer dependent on layer order.

Description

True if this layer has track matte. When true, this layer's `trackMatteType` value controls how the matte is applied. See *AVLayer.trackMatteType* for available track matte types.

Type

Boolean; read-only.

18.1.15 AVLayer.height

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

Description

The height of the layer in pixels.

Type

Floating-point; read-only.

18.1.16 AVLayer.isNameFromSource

`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.

18.1.17 AVLayer.isTrackMatte

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

Note: This functionality was updated in After Effects 23.0. Track Matte is no longer dependent on layer order.

Description

True if this layer is being used as a track matte.

Type

Boolean; read-only.

18.1.18 AVLayer.motionBlur

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

Description

True if motion blur is enabled for the layer.

Type

Boolean; read/write.

18.1.19 AVLayer.preserveTransparency

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

Description

True if preserve transparency is enabled for the layer.

Type

Boolean; read/write.

18.1.20 AVLayer.quality

`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`

18.1.21 AVLayer.samplingQuality

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

Note: This functionality was added in After Effects 12.0 (CC)

Description

Set/get layer sampling method (bicubic or bilinear)

Type

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

- `LayerSamplingQuality.BICUBIC`
- `LayerSamplingQuality.BILINEAR`

18.1.22 AVLayer.source

`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.

18.1.23 AVLayer.threeDLayer

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

Description

True if this is a 3D layer.

Type

Boolean; read/write.

18.1.24 AVLayer.threeDPerChar

`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.

18.1.25 AVLayer.timeRemapEnabled

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

Description

True if time remapping is enabled for this layer.

Type

Boolean; read/write.

18.1.26 AVLayer.trackMatteLayer

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

Note: This functionality was added in After Effects 23.0

Description

Returns the track matte layer for this layer. Returns `null` if this layer has no track matte layer.

Type

AVLayer object; read only.

18.1.27 AVLayer.trackMatteType

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

Note: This functionality was updated in After Effects 23.0

Warning: This is a Legacy API we don't recommend using for setting Track Matte Type in new scripts. Please consider using the latest track matte APIs *AVLayer.setTrackMatte()* and *AVLayer.removeTrackMatte()* for your tasks.

Description

If this layer has a track matte, specifies the way the track matte is applied. Specifying the `TrackMatteType.NO_TRACK_MATTE` type will remove the track matte for this layer and reset the track matte type.

Type

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

- `TrackMatteType.ALPHA`

- TrackMatteType.ALPHA_INVERTED
- TrackMatteType.LUMA
- TrackMatteType.LUMA_INVERTED
- TrackMatteType.NO_TRACK_MATTE

Example

```
// Returns the current track matte type for myLayer
var type = myLayer.trackMatteType;

// *** We recommend using the new Track Matte APIs for the operations below (See␣
↪Warning) ***

// Changes the track matte type for myLayer to TrackMatteType.ALPHA_INVERTED
myLayer.trackMatteType = TrackMatteType.ALPHA_INVERTED;

// Removes the track matte and also resets the type
myLayer.trackMatteType = TrackMatteType.NO_TRACK_MATTE;
```

18.1.28 AVLayer.width

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

Description

The width of the layer in pixels.

Type

Floating-point; read-only.

18.2 Methods

18.2.1 AVLayer.addToMotionGraphicsTemplate()

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

Note: This functionality was added in After Effects 18.0 (2021)

Description

Adds the layer to the Essential Graphics Panel for the specified composition.

Returns true if the layer is successfully added, or false otherwise.

- If the layer cannot be added, it is either because it is not a layer type for which media can be replaced (referred to as Media Replacement Layers), or the layer has already been added to the EGP for that composition. After Effects will present a warning dialog if the layer cannot be added to the EGP.

- Use the *AVLayer.canAddToMotionGraphicsTemplate()* method to test whether the layer can be added to a Motion Graphics template.

Parameters

`comp`	A CompItem object; the composition where you wish to add the property to the EGP. Required.

Returns

Boolean.

18.2.2 AVLayer.addToMotionGraphicsTemplateAs()

```
app.project.item(index).layer(index).addToMotionGraphicsTemplateAs(comp, name)
```

Note: This functionality was added in After Effects 18.0 (2021)

Description

Adds the layer to the Essential Graphics Panel for the specified composition.

Returns true if the layer is successfully added, or false otherwise.

- If the layer cannot be added, it is either because it is not a layer type for which media can be replaced (referred to as Media Replacement Layers), or the layer has already been added to the EGP for that composition. After Effects will present a warning dialog if the layer cannot be added to the EGP.
- Use the *AVLayer.canAddToMotionGraphicsTemplate()* method to test whether the layer can be added to a Motion Graphics template.

Parameters

`comp`	A CompItem object; the composition where you wish to add the property to the EGP. Required.
`name`	A string for the new name. Required.

Returns

Boolean.

18.2.3 AVLayer.audioActiveAtTime()

```
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.

18.2.4 AVLayer.calculateTransformFromPoints()

`app.project.item(index).layer(index).calculateTransformFromPoints(pointTopLeft, pointTopRight, pointBottomRight)`

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]);
}
```

18.2.5 AVLayer.canAddToMotionGraphicsTemplate()

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

Note: This functionality was added in After Effects 18.0 (2021)

Description

Test whether or not the layer can be added to the Essential Graphics Panel for the specified composition.

Returns true if the layer can be added, or false otherwise.

If the layer cannot be added, it is either because it is not a layer type for which media can be replaced (referred to as Media Replacement Layers), or the layer has already been added to the EGP for that composition.

Media Replacement layers are recognized as AVLayers with an *AVLayer.source* set to a *FootageItem object* (with specific source types) or a *CompItem object*.

The AVLayer needs to comply with the restrictions below in order to be treated as a Media Replacement layer:

- *Layer.hasVideo* should return true.
- *AVLayer.adjustmentLayer* should return false.
- *Layer.nullLayer* should return false.
- If the *AVLayer.source* is a *FootageItem object*, then FootageItem.FootageSource should not be a *SolidSource object*.
- If the *AVLayer.source* is a *FootageItem object* and the FootageItem.FootageSource is a *FileSource object* then that FileSource should not point to a non-media file e.g. a JSX script file.

Parameters

`comp`	A CompItem object; the composition where you wish to add the property to the EGP. Required.

Returns

Boolean.

18.2.6 AVLayer.compPointToSource()

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

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

Converts composition coordinates, such as `sourcePointToComp`, to layer coordinates.

Warning: This value only reflects the first character in the text layer at the current time.

Parameters

`sourcePointToComp`	A position array of composition coordinates in ([X, Y]) format.

Returns

Array of ([X,Y]) position coordinates; read-only.

18.2.7 AVLayer.openInViewer()

```
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).

18.2.8 AVLayer.removeTrackMatte()

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

Note: This functionality was added in After Effects 23.0

Description

Removes the track matte for this layer while preserving the TrackMatteType. See *AVLayer.setTrackMatte()* for another way of removing track matte.

Parameters

None.

Returns

Nothing.

```
// Sets the track matte layer of myLayer with otherLayer as LUMA type
myLayer.setTrackMatte(otherLayer, TrackMatteType.LUMA);

// Removes the track matte for myLayer but preserves the LUMA type
myLayer.removeTrackMatte();

// Still returns TrackMatteType.LUMA
alert(myLayer.trackMatteType);
```

18.2.9 AVLayer.replaceSource()

`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()*.

Returns

Nothing.

Warning: If this method is performed on a null layer, the layers `isNull` attribute is not changed from `true`. This causes the layer not to be visible in comp viewer and renders.

18.2.10 AVLayer.setTrackMatte()

`app.project.item(index).layer(index).setTrackMatte(trackMatteLayer, trackMatteType)`

Note: This functionality was added in After Effects 23.0

Description

Sets the track matte layer and type for this layer. Passing in `null` to trackMatteLayer parameter removes the track matte. See *AVLayer.removeTrackMatte()* for another way of removing track matte.

Parameters

`trackMatteLayer`	The AVLayer to be used as the track matte layer.
`trackMatteType`	The type of the track matte to be used. Please see *AVLayer.trackMatteType* for available types.

Warning: Passing in `TrackMatteType.NO_TRACK_MATTE` as type is invalid and will result in no-op.

Returns

Nothing

Example

```
// Sets the track matte layer of myLayer with otherLayer as Alpha type
myLayer.setTrackMatte(otherLayer, TrackMatteType.ALPHA);

// Keeps the same trackMatteLayer and only changes the track matte type
myLayer.setTrackMatte(myLayer.trackMatteLayer, TrackMatteType.LUMA);

// Changes the track matte layer but keep the same track matte type
myLayer.setTrackMatte(newTrackMatteLayer, myLayer.trackMatteType);

// Removes the track matte for myLayer and sets the new specified TrackMatteType
myLayer.setTrackMatte(null, TrackMatteType.ALPHA);
myLayer.setTrackMatte(null, TrackMatteType.NO_TRACK_MATTE);

// Invalid. Nothing happens
myLayer.setTrackMatte(otherLayer, TrackMatteType.NO_TRACK_MATTE);
```

18.2.11 AVLayer.sourcePointToComp()

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

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

Converts layer coordinates, such as `boxTextPos`, to composition coordinates.

Warning: This value only reflects the first character in the text layer at the current time.

Parameters

`boxTextPos`	A position array of layer coordinates in ([X, Y]) format.

Returns

Array of ([X,Y]) position coordinates; read-only.

Example

```
// For a paragraph text layer.
// Converts position in layer coordinates to comp coordinates.
var boxTextCompPos = myTextLayer.sourcePointToComp(boxTextLayerPos);
```

18.2.12 AVLayer.sourceRectAtTime()

`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`].

CHAPTER

NINETEEN

CAMERALAYER OBJECT

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

Description

The CameraLayer object represents a camera layer within a composition. Create it using *LayerCollection.addCamera()*. 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 object*. All methods and attributes of Layer are available when working with CameraLayer.

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`
  - `PointofInterest`
  - `Position`
  - `Scale`
  - `Orientation`
  - `XRotation`
  - `YRotation`
  - `Rotation`
  - `Opacity`
- `CameraOptions`
  - `Zoom`
  - `DepthofField`
  - `FocusDistance`
  - `BlurLevel`

CHAPTER

TWENTY

LIGHTLAYER OBJECT

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

Description

The LightLayer object represents a light layer within a composition. Create it using the *LayerCollection.addLight()* method. 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 object*. All methods and attributes of Layer are available when working with Light-Layer.

AE Properties

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

- `Marker`
- **`Transform`**
  - `PointofInterest`
  - `Position`
  - `Scale`
  - `Orientation`
  - `XRotation`
  - `YRotation`
  - `Rotation`
  - `Opacity`
- **`LightOptions`**
  - `Intensity`
  - `Color`
  - `ConeAngle`
  - `ConeFeather`
  - `CastsShadows`
  - `ShadowDarkness`
  - `ShadowDiffusion`

20.1 Attributes

20.1.1 LightLayer.lightType

`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`

CHAPTER

TWENTYONE

SHAPELAYER OBJECT

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

Description

The ShapeLayer object represents a shape layer within a composition. Create it using *LayerCollection.addShape()*. 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.

CHAPTER

TWENTYTWO

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*. 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.

AE Properties

TextLayer defines no additional attributes, but has the following AE properties and property groups, in addition to those inherited from AVLayer:

- `Text`
- `SourceText`
- `PathOptions`
- `Path`
- `ReversePath`
- `PerpendicularToPath`
- `ForceAlignment`
- `FirstMargin`
- `LastMargin`
- `MoreOptions`
- `AnchorPointGrouping`
- `GroupingAlignment`
- `Fill&Stroke`
- `InterCharacterBlending`
- `Animators`

Unused Properties and Attributes

The `TimeRemapandMotionTrackers` properties, inherited from AVLayer, are not applicable to text layers, and their related AVLayer attributes are not used:

- `canSetTimeRemapEnabled`
- `timeRemapEnabled`

- trackMatteType
- isTrackMatte
- hasTrackMatte

CHAPTER

TWENTYTHREE

THREEDMODELLAYER OBJECT

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

Note: This functionality was added in After Effects (Beta) 24.4 and is subject to change while it remains in Beta.

Description

The ThreeDModelLayer object represents a 3D Model layer within a composition.

ThreeDModelLayer is a subclass of *AVLayer object*. All methods and attributes of AVLayer are available when working with ThreeDModelLayer.

AE Properties

ThreeDModelLayer inherits the following properties and property groups from *AVLayer object*:

- Marker
- Time Remap
- Transform
  - Anchor Point
  - Position
  - Scale
  - Orientation
  - X Rotation
  - Y Rotation
  - Rotation
  - Opacity
- Layer Styles
- Audio
  - AudioLevels

Example

If the first item in the project is a CompItem, and the first layer of that CompItem is an ThreeDModelLayer, the following checks its type.

```
var modelLayer = app.project.item(1).layer(1);
if (modelLayer instanceof ThreeDModelLayer)
{
   // do something
}
```

CHAPTER

TWENTYFOUR

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("AddGrain")("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()*.

> PropertyBase is the base class for both *Property* and *PropertyGroup*, so PropertyBase attributes and methods are available when working with properties and property groups.

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 anobject, 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 inselected object
```

A less straightforward case is when a property is removed from a property group. In this case, After Effectsgenerates 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
```

24.1 Attributes

24.1.1 PropertyBase.active

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

Description

For a layer, this corresponds to the setting of the eyeball icon. 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 *AVLayer.audioActive*.

For an effect and all properties, it is the same as the enabled attribute, except that it's read-only.

Type

Boolean; read-only.

24.1.2 PropertyBase.canSetEnabled

```
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.

24.1.3 PropertyBase.elided

```
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`
- `PathOptions`
- `MoreOptions`
- `Animator1`

- `Animator2`

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.

24.1.4 PropertyBase.enabled

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

Description

For layer, this corresponds to the video switch state of the layer in the Timeline panel. For an effect and all properties, it corresponds to the setting of the eyeball icon, if there is one.

When true, the layer or property is enabled; otherwise false.

Type

Boolean; read/write if `canSetEnabled` is true, read-only if `canSetEnabled` is false.

24.1.5 PropertyBase.isEffect

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

Description

When true, this property is an effect PropertyGroup.

Type

Boolean; read-only.

24.1.6 PropertyBase.isMask

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

Description

When true, this property is a mask PropertyGroup.

Type

Boolean; read-only.

24.1.7 PropertyBase.isModified

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

Description

When true, this property has been changed since its creation.

Type

Boolean; read-only.

24.1.8 PropertyBase.matchName

`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*.)

Type

String; read-only.

24.1.9 PropertyBase.name

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

Description

For a layer, the name of the layer. By default, this is the same as the Source name, unless *Layer.isNameSet* returns false.

For an effect and all properties - the display name of the property. (Compare *PropertyBase.matchName*.) 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*).

Type

String; read/write for a child of an indexed group; otherwise read-only.

24.1.10 PropertyBase.parentProperty

`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.

24.1.11 PropertyBase.propertyDepth

`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.

24.1.12 PropertyBase.propertyIndex

`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*).

Type

Integer; read-only.

24.1.13 PropertyBase.propertyType

`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.

24.1.14 PropertyBase.selected

`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 either *CompItem.selectedProperties* or *Layer.selectedProperties*.

Type

Boolean; read/write.

24.2 Methods

24.2.1 PropertyBase.duplicate()

`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*.

Parameters

None.

Returns

PropertyBase object.

24.2.2 PropertyBase.moveTo()

`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 `PropertyType.INDEXED_GROUP`; see *PropertyBase.propertyType*.)

> **Warning:** 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.

24.2.3 PropertyBase.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.

24.2.4 PropertyBase.remove()

`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*.) 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.

CHAPTER

TWENTYFIVE

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 a value, often animatable, of an effect, mask, or transform within an individual layer. For examples of how to access properties, see *PropertyBase object* and *PropertyGroup.property()*.

> Property is a subclass of *PropertyBase*. All methods and attributes of PropertyBase, in addition to those listed below, are available when working with Property.

Note: JavaScript objects commonly referred to as "properties" are called "attributes" in this guide, to avoid confusion with the After Effects definition of property.

Examples

- 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 my Opacity is a float value
var myOpacity = myProperty.value;
```

- 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 my Position is an array of 3 floats
var myPosition = myProperty.value;
```

- 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);
```

- 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);
```

- 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");
}
```

- 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

```
var myProperty = myLayer.rotation;
myProperty.setValueAtTime(0, 0);
myProperty.setValueAtTime(5, 90);
myProperty.setValueAtTime(10, 0);
```

- Change the key frame values for the first three keyframes of some sourcetext

```
var myProperty = myTextLayer.sourceText;
if (myProperty.numKeys < 3) {
    alert("error, I thought there were 3 keyframes");
} else {
    myProperty.setValueAtKey(1, newTextDocument("keynumber1"));
    myProperty.setValueAtKey(2, newTextDocument("keynumber2"));
    myProperty.setValueAtKey(3, newTextDocument("keynumber3"));
}
```

- 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 defaultof 100.
myLayer.scale.setValue([50, 50, 100]);
myLayer.scale.setValue([50, 50]);
// These two are equivalent. The second fills in a defaultof 1.0
myLight.color.setValue([0.8, 0.3, 0.1, 1.0]);
myLight.color.setValue([0.8, 0.3, 0.1]);
// These two are equivalent. The second creates a TextDocument
myTextLayer.sourceText.setValue(newTextDocument("foo"));
myTextLayer.sourceText.setValue("foo");
```

25.1 Attributes

25.1.1 Property.alternateSource

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

Note: This functionality was added in After Effects 18.0 (2021)

Description

The value is null when:

- The alternate source is not set for the associated layer.
- The property cannot be used to set an alternate source.

Use *Property.canSetAlternateSource* to determine if the property is a Media Replacement Essential Property.

All Media Replacement Layers have an alternate source item that can be set.

A layer is "marked" for media replacement when the layer is added to the Essential Graphics Panel (see *AVLayer.addToMotionGraphicsTemplate()* or *AVLayer.addToMotionGraphicsTemplateAs()*).

- If present, the render workflow will pick up the alternate source while rendering the layer.
- If the alternate source for the layer is not set, then the source layer of the Media Replacement control is used for rendering (this is the normal workflow).

Use *Property.setAlternateSource()* to change the value.

Type

AVItem object; read-only.

25.1.2 Property.canSetAlternateSource

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

Note: This functionality was added in After Effects 18.0 (2021)

Description

Test whether the property is an Essential Property that supports Media Replacement.

Returns true if the property allows Media Replacement, false otherwise.

Type

Boolean; read-only.

25.1.3 Property.canSetExpression

`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.

Type

Boolean; read-only.

25.1.4 Property.canVaryOverTime

`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.

25.1.5 Property.dimensionsSeparated

`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.

25.1.6 Property.essentialPropertySource

`app.project.item(index).layer(index).essentialProperty.property(index).essentialPropertySource`

Note: This functionality was added in After Effects 22.0 (2022)

Description

Instance property on an Essential Property object which returns the original source Property which was used to create the Essential Property.

Type

Can be either:

- A read/write *Property object*, in the case that the source object used to create the Essential Property was a Property
- A read/write *AVLayer object*, in the case that the source object used to create the Essential Property was a Media Replacement Footage item
- Null if called on a non-Essential Property

Example

```
var firstComp = app.project.item(1);
var opacityProp = firstComp.layer(1).property("Transform").property("Opacity");

opacityProp.addToMotionGraphicsTemplate(firstComp);

var secondComp = app.project.item(2);
secondComp.layers.add(firstComp);

var essentialOpacity = secondComp.layer(1).essentialProperty.property(1);
if (essentialOpacity.essentialPropertySource == opacityProp) {
  alert("You can get the source Property from an Essential Property!");
}
```

25.1.7 Property.expression

`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.

25.1.8 Property.expressionEnabled

`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.

25.1.9 Property.expressionError

`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.

25.1.10 Property.hasMax

`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.

25.1.11 Property.hasMin

`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.

25.1.12 Property.isDropdownEffect

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

Note: This functionality was added in After Effects 17.0.1 (2020)

Description

When true, the property is the Menu property of a Dropdown Menu Control effect and can have its items updated with *setPropertyParameters*.

Examples

```
appliedEffect.property("Menu").isDropdownEffect;    // true
appliedEffect.property("Color").isDropdownEffect;   // false
appliedEffect.property("Feather").isDropdownEffect; // false
```

Type

Boolean; read-only.

25.1.13 Property.isSeparationFollower

`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.

25.1.14 Property.isSeparationLeader

`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.

25.1.15 Property.isSpatial

`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.

25.1.16 Property.isTimeVarying

`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.

25.1.17 Property.maxValue

`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.

25.1.18 Property.minValue

`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.

25.1.19 Property.numKeys

`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.

25.1.20 Property.propertyIndex

`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.

25.1.21 Property.propertyValueType

`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*
- `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*
- `PropertyValueType.TEXT_DOCUMENT`: *TextDocument object*

25.1.22 Property.selectedKeys

`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.

25.1.23 Property.separationDimension

`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.

25.1.24 Property.separationLeader

`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.

25.1.25 Property.unitsText

`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.

25.1.26 Property.value

`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*.

Type

A value appropriate for the type of the property (see *Property.propertyValueType*); read-only.

25.2 Methods

25.2.1 Property.addKey()

`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.

25.2.2 Property.addToMotionGraphicsTemplate()

`app.project.item(index).layer(index).propertySpec.addToMotionGraphicsTemplate(comp)`

Note: This functionality was added in After Effects 15.0 (CC 2018)

Description

Adds the property to the Essential Graphics panel for the specified composition.

Returns true if the property is successfully added, false otherwise.

If the property is not added, it is either because it is not one of the supported property types or the property has already been added to the EGP for that composition. After Effects will present a warning dialog if the property cannot be added to the EGP.

Use the *Property.canAddToMotionGraphicsTemplate()* method to test whether the property can be added to a Motion Graphics template.

Parameters

`comp`	The composition that you wish to add the property to, a CompItem. Required.

Returns

Boolean.

25.2.3 Property.addToMotionGraphicsTemplateAs()

```
app.project.item(index).layer(index).propertySpec.addToMotionGraphicsTemplateAs(comp,
name)
```

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

Adds the property to the Essential Graphics panel for the specified composition, but with an additional option to give the EGP property a custom name.

Returns true if the property is successfully added, false otherwise.

If the property is not added, it is either because it is not one of the supported property types or the property has already been added to the EGP for that composition. After Effects will present a warning dialog if the property cannot be added to the EGP.

Use the *Property.canAddToMotionGraphicsTemplate()* method to test whether the property can be added to a Motion Graphics template.

Parameters

`comp`	The composition that you wish to add the property to, a CompItem. Required.
`name`	A string for the new name. Required.

Returns

Boolean.

25.2.4 Property.canAddToMotionGraphicsTemplate()

```
app.project.item(index).layer(index).propertySpec.canAddToMotionGraphicsTemplate(comp)
```

Note: This functionality was added in After Effects 15.0 (CC 2018)

Description

Test whether or not the property can be added to the Essential Graphics panel for the specified composition.

Returns true if the property can be added, false otherwise.

If the property can not be added, it is either because it is not one of the supported property types or the property has already been added to the EGP for that composition. After Effects will present a warning dialog if the property cannot be added to the EGP.

Supported property types are:

- Checkbox
- Color
- Numerical Slider (i.e., a single-value numerical property, such as Transform > Opacity or the Slider Control expression control effect)
- Source Text

Parameters

`comp`	The composition that you wish to add the property to, a CompItem. Required.

Returns

Boolean.

25.2.5 Property.getSeparationFollower()

`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.

25.2.6 Property.isInterpolationTypeValid()

`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.

25.2.7 Property.keyInInterpolationType()

`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*.

Returns

A `KeyframeInterpolationType` enumerated value; one of:

- `KeyframeInterpolationType.LINEAR`
- `KeyframeInterpolationType.BEZIER`
- `KeyframeInterpolationType.HOLD`

25.2.8 Property.keyInSpatialTangent()

`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_SPATIALorThreeD_SPATIAL`).

Parameters

`keyIndex`	The index for the keyframe. An integer in the range `[1..numKeys]`, as returned by the *addKey* or *nearestKeyIndex*.

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.

25.2.9 Property.keyInTemporalEase()

`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 *nearestKeyIndex*.

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.

25.2.10 Property.keyLabel()

`app.project.item(index).layer(index).propertySpec.keyLabel(keyIndex)`

Note: This functionality was added in After Effects 22.6.

Description

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

Read only. Keyframe color labels can be set by *setLabelAtKey*.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range `[1..numKeys]`, as returned by the *addKey* or *nearestKeyIndex*.

Returns

Integer (0 to 16); read only.

25.2.11 Property.keyOutInterpolationType()

`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 *nearestKeyIndex*.

Returns

A `KeyframeInterpolationType` enumerated value; one of:

- `KeyframeInterpolationType.LINEAR`
- `KeyframeInterpolationType.BEZIER`
- `KeyframeInterpolationType.HOLD`

25.2.12 Property.keyOutSpatialTangent()

`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 *nearestKeyIndex*.

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.

25.2.13 Property.keyOutTemporalEase()

`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 *nearestKeyIndex*.

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.

25.2.14 Property.keyRoving()

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*.

Returns

Boolean.

25.2.15 Property.keySelected()

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*.

Returns

Boolean.

25.2.16 Property.keySpatialAutoBezier()

`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*.

Returns

Boolean.

25.2.17 Property.keySpatialContinuous()

`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*.

Returns

Boolean.

25.2.18 Property.keyTemporalAutoBezier()

`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*.

Returns

Boolean.

25.2.19 Property.keyTemporalContinuous()

`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*.

Returns

Boolean.

25.2.20 Property.keyTime()

`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*.
`markerComment`	The comment string attached to a marker (see *MarkerValue.comment* attribute).

Returns

Floating-point value.

25.2.21 Property.keyValue()

`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*.
`markerComment`	The comment string attached to a marker (see *MarkerValue.comment* attribute).

Returns

Floating-point value for keyframes, MarkerValue object for markers.

25.2.22 Property.nearestKeyIndex()

`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.

25.2.23 Property.removeKey()

`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 *nearestKeyIndex*.

Returns

Nothing.

25.2.24 Property.setAlternateSource()

```
app.project.item(index).layer(index).propertySpec.setAlternateSource(newSource)
```

Note: This functionality was added in After Effects 18.0 (2021)

Description

Set the alternate source for this property.

The Property object and the input parameters for the AVItem that is being called needs to be Media Replacement compatible for the action to go through.

- Use the *AVItem.isMediaReplacementCompatible* method to test whether the AVItem can be used as an alternate source for Media Replacement.
- Use *Property.canSetAlternateSource* to test if the property allows Media Replacement.

Parameters

`newSource`	The new source AVItem object. Required.

Returns

Nothing.

25.2.25 Property.setInterpolationTypeAtKey()

```
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*.
`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.

25.2.26 Property.setLabelAtKey()

```
app.project.item(index).layer(index).propertySpec.setLabelAtKey(keyIndex, labelIndex)
```

Note: This functionality was added in After Effects 22.6 (2022)

Description

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

Parameters

`keyIndex`	The index for the keyframe. An integer in the range `[1..numKeys]`, as returned by the *addKey* or *nearestKeyIndex*.
`labelIndex`	The index for the new label value. An integer in the range 0-16.

Returns

Nothing.

25.2.27 Property.setPropertyParameters()

`app.project.item(index).layer(index).propertySpec.setPropertyParameters(items)`

Note: This functionality was added in After Effects 17.0.1 (2020)

Description

Sets parameters for a Dropdown Menu Control's Menu Property. This method will overwrite the existing set of Menu items with the provided array of strings.

- The Dropdown Menu Control effect's Menu property is the only property that allows parameters to be set.
- To check if a property allows parameters to be set, check with *isDropdownEffect* before calling this method.
- An exception is raised whenever this method fails.

Parameters

items	An array of strings which will replace the existing menu entries in a Dropdown Menu Control. • Only strings are allowed. • Empty item strings are not allowed. • Duplicate item strings are not allowed. • The character "|" is not allowed in the item strings. • The string "(-" - can be specified as of the item strings. These appear as separator lines in the dropdown menu. The separator lines will claim an index for each of themselves.

Note: Item strings should be in ASCII or MultiByte encodable in the current code-page. In other words, the item strings should be provided in the script of the running system. For example: Specifying the item strings in Japanese while running the script on an English system will create a dropdown effect with illegible characters in the item strings.

Example

```
var dropdownItems = [
    "First Item",
    "Second Item",
    "(-",
    "Another Item",
    "Last Item"
];

var dropdownEffect = layer.property("ADBE Effect Parade").addProperty("ADBE Dropdown␣
↪Control");
dropdownEffect.property(1).setPropertyParameters(dropdownItems);
```

Returns

Property object, the updated Dropdown Menu Control's Menu property.

25.2.28 Property.setRovingAtKey()

`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 *nearestKeyIndex*.
`newVal`	True to turn roving on, false to turn roving off.

Returns

Nothing.

25.2.29 Property.setSelectedAtKey()

`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 *nearestKeyIndex*.
`onOff`	True to select the keyframe, false to deselect it.

Returns

Nothing.

25.2.30 Property.setSpatialAutoBezierAtKey()

`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 *nearestKeyIndex*.
newVal	True to turn spatial auto-Bezier on, false to turn it off.

Returns

Nothing.

25.2.31 Property.setSpatialContinuousAtKey()

`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 *nearestKeyIndex*.
newVal	True to turn spatial auto-Bezier on, false to turn it off.

Returns

Nothing.

25.2.32 Property.setSpatialTangentsAtKey()

`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 *nearestKeyIndex()* 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.

25.2.33 Property.setTemporalAutoBezierAtKey()

`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 *nearestKeyIndex*.
newVal	True to turn temporal auto-Bezier on, false to turn it off.

Returns

Nothing.

25.2.34 Property.setTemporalContinuousAtKey()

`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 *nearestKeyIndex*.
`newVal`	True to turn temporal continuity on, false to turn it off.

Returns

Nothing.

25.2.35 Property.setTemporalEaseAtKey()

`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*.

Parameters

`keyIndex`	The index for the keyframe. An integer in the range `[1. .numKeys]`, as returned by the *addKey* or *nearestKeyIndex*.
`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.

25.2.36 Property.setValue()

`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()* or *Property.setValueAtKey()*.

Parameters

`newValue`	A value appropriate for the type of property being set; see *Property.propertyValueType*.

Returns

Nothing.

25.2.37 Property.setValueAtKey()

`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 *nearestKeyIndex*.
`newValue`	A value appropriate for the type of property being set; see *Property.propertyValueType*.

Returns

Nothing.

25.2.38 Property.setValueAtTime()

`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*.

Returns

Nothing.

25.2.39 Property.setValuesAtTimes()

`app.project.item(index).layer(index).propertySpec.setValuesAtTimes(times, newValues)`

Description

Sets values for a set of keyframes at specified 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 composition is 0.
`newValues`	A array of values appropriate for the type of property being set; see *Property.propertyValueType*.

Returns

Nothing.

25.2.40 Property.valueAtTime()

`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.

Note: As After Effects 13.6, this method now waits for time-intensive expressions, like `sampleImage`, to finish evaluating before it returns the result.

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).

CHAPTER

TWENTYSIX

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()*. For examples of how to access properties and property groups, see *PropertyBase object*.

> PropertyGroup is a subclass of *PropertyBase*. All methods and attributes of PropertyBase, in addition to those listed below, are available when working with PropertyGroup.
>
> PropertyGroup is a base class for *Layer* and *MaskPropertyGroup*. PropertyGroup attributes and methods are available when working with layer or mask groups.

26.1 Attributes

26.1.1 PropertyGroup.numProperties

`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 *PropertyGroup.property()*.

Type

Integer; read-only.

26.2 Methods

26.2.1 PropertyGroup.addProperty()

`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*). 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 *PropertyGroup.canAddProperty()*.)

> **Warning:** When you add a new property to an indexed group, the indexed group gets recreated from scratch, invalidating all existing references to properties.
>
> One workaround is to store the index of the added property with *property.propertyIndex*.

Examples

- This won't work, as the *slider* object becomes invalid once we add the *Color Control* property:

```
var effectsProperty = layer.property("ADBE Effect Parade");
var slider = effectsProperty.addProperty("ADBE Slider Control");
var color = effectsProperty.addProperty("ADBE Color Control");

var sliderProperty = slider.property("ADBE Slider Control-0001"); //␣
↪Object 'slider' is Invalid
```

- This revised method will work:

```
var effectsProperty = layer.property("ADBE Effect Parade");
var slider = effectsProperty.addProperty("ADBE Slider Control");
var sliderIndex = slider.propertyIndex; // Store 'slider' effect index so␣
↪it can be reused later
var color = effectsProperty.addProperty("ADBE Color Control");

var sliderProperty = effectsProperty.property(sliderIndex).property("ADBE␣
↪Slider Control-0001");
```

Parameters

name	The display name or match name of the property to add. (See *PropertyBase.matchName*). 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.

26.2.2 PropertyGroup.canAddProperty()

`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()*.

Returns

Boolean.

26.2.3 PropertyGroup.property()

```
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 [1..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)
26.2. Methods	• “Casts Shadows” or “castsShadows” **209** • “Light Transmission” or “lightTransmission” • “Metal” or “metal”

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("Mask1");
```

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");
```

CHAPTER

TWENTYSEVEN

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 object*. All methods and attributes of *PropertyBase object* and PropertyGroup, in addition to those listed below, are available when working with MaskPropertyGroup.

27.1 Attributes

27.1.1 MaskPropertyGroup.color

`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.

27.1.2 MaskPropertyGroup.inverted

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

Description

When true, the mask is inverted; otherwise false.

Type

Boolean; read/write.

27.1.3 MaskPropertyGroup.locked

`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.

27.1.4 MaskPropertyGroup.maskFeatherFalloff

`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`

27.1.5 MaskPropertyGroup.maskMode

`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`

27.1.6 MaskPropertyGroup.maskMotionBlur

`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`

27.1.7 MaskPropertyGroup.rotoBezier

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

Description

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

Type

Boolean; read/write.

CHAPTER

TWENTYEIGHT

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.

28.1 Attributes

28.1.1 RenderQueue.canQueueInAME

`app.project.renderQueue.canQueueInAME`

Note: This functionality was added in After Effects 14.0 (CC 2017)

Description

indicates whether or not there are queued render items in the After Effects render queue. Only queued items can be added to the AME queue.

RenderQueue.queueInAME()

Type

Boolean; read-only.

28.1.2 RenderQueue.queueNotify

`app.project.renderQueue.queueNotify`

Note: This functionality was added in After Effects 22.0 (2022)

Description

Read or write the **Notify** property for the entire Render Queue. This is exposed in the UI as a checkbox in the lower right corner of the Render Queue panel.

Type

Boolean; read/write.

28.1.3 RenderQueue.items

`app.project.renderQueue.items`

Description

A collection of all items in the render queue. See *RenderQueueItem object*.

Type

RQItemCollection object; read-only.

28.1.4 RenderQueue.numItems

`app.project.renderQueue.numItems`

Description

The total number of items in the render queue.

Type

Integer; read-only.

28.1.5 RenderQueue.rendering

`app.project.renderQueue.rendering`

Description

When true, the rendering process is in progress or paused. When false, it is stopped.

Type

Boolean; read-only.

28.2 Methods

28.2.1 RenderQueue.item()

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

Description

Gets a specified item from the ite ms collection.

Parameters

`index` The position index of the item. An integer in the range `[0..numItems]`.

Returns

RenderQueueItem object.

28.2.2 RenderQueue.pauseRendering()

`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 *RenderQueueItem.onstatus* or *app.onError* callback.

Parameters

`pause`	True to pause a current render process, false to continue a paused render.

Returns

Nothing.

28.2.3 RenderQueue.render()

`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 *RenderQueue.pauseRendering()* or *RenderQueue.stopRendering()* from an `onError` or `onstatus` callback.

- To respond to errors during the rendering process, define a callback function in *app.onError*.
- To respond to changes in the status of a particular item while the render is progressing, define a callback function in *RenderQueueItem.onstatus* in the associated RenderQueueItem object.

Parameters

None.

Returns

Nothing.

28.2.4 RenderQueue.showWindow()

`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.

28.2.5 RenderQueue.stopRendering()

`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 *RenderQueueItem.onstatus* or *app.onError* callback.

Parameters

None.

Returns

Nothing.

28.2.6 RenderQueue.queueInAME()

`app.project.renderQueue.queueInAME(render_immediately_in_AME)`

Note: This functionality was added in After Effects 14.0 (CC 2017)

Description

Calls the Queue In AME command. This method requires passing a boolean value, telling AME whether to only queue the render items (false) or if AME should also start processing its queue (true).

Note: This requires Adobe Media Encoder CC 2017 (11.0) or later.

Note: When AME receives the queued items, it applies the most recently used encoding preset. If `render_immediately_in_AME` is set to true, you will not have an opportunity to change the encoding settings.

Parameters

`render_immediately_in_AME`	Telling AME whether to only queue the render items (`false`) or if AME should also start processing its queue (`true`).

Returns

Nothing.

Example

The following sample code checks to see if there are queued items in the render queue, and if so queues them in AME but does not immediately start rendering:

```
// Scripting support for Queue in AME.
// Requires Adobe Media Encoder 11.0.
if (app.project.renderQueue.canQueueInAME === true) {
    // Send queued items to AME, but do not start rendering.
    app.project.renderQueue.queueInAME(false);
} else {
    alert("There are no queued item in the Render Queue.");
}
```

CHAPTER

TWENTYNINE

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.

> RQItemCollection is a subclass of *Collection object*. All methods and attributes of Collection are available when working with RQItemCollection.

29.1 Methods

29.1.1 RQItemCollection.add()

`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.

CHAPTER

THIRTY

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()*.

30.1 Attributes

30.1.1 RenderQueueItem.comp

`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.

30.1.2 RenderQueueItem.elapsedSeconds

`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.

30.1.3 RenderQueueItem.logType

`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`

30.1.4 RenderQueueItem.numOutputModules

`app.project.renderQueue.item(index).numOutputModules`

Description

The total number of Output Modules assigned to this item.

Type

Integer; read-only.

30.1.5 RenderQueueItem.onstatus

`app.project.renderQueue.item(index).onstatus`

Description

The name of a callback function that is called whenever the value of the *RenderQueueItem.status* attribute changes.

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()* and *RenderQueue.stopRendering()*. See also *app.onError*.

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).onstatus = myStatusChanged();
app.project.renderQueue.item(1).render = false; // changes status and shows dialog
```

30.1.6 RenderQueueItem.outputModules

`app.project.renderQueue.item(index).outputModules`

Description

The collection of Output Modules for the item.

Type

OMCollection object; read-only.

30.1.7 RenderQueueItem.queueItemNotify

`app.project.renderQueue.item(index).queueItemNotify`

Note: This functionality was added in After Effects 22.0 (2022)

Description

Scripts can read and write the **Notify** checkbox for each individual item in the Render Queue. This is exposed in the UI as a checkbox next to each Render Queue item in the Notify column.

This column is hidden by default and may need to be selected to be visible by right clicking on the Render Queue column headers and choosing Notify.

Type

Boolean; read/write.

30.1.8 RenderQueueItem.render

`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.

30.1.9 RenderQueueItem.skipFrames

`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.

30.1.10 RenderQueueItem.startTime

`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.

30.1.11 RenderQueueItem.status

`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.

30.1.12 RenderQueueItem.templates

`app.project.renderQueue.item(index).templates`

Description

The names of all Render Settings templates available for the item. See also *RenderQueueItem.saveAsTemplate()*.

Type

Array of strings; read-only.

30.1.13 RenderQueueItem.timeSpanDuration

`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.

30.1.14 RenderQueueItem.timeSpanStart

`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.

30.2 Methods

30.2.1 RenderQueueItem.applyTemplate()

`app.project.renderQueue.item(index).applyTemplate(templateName)`

Description

Applies a Render Settings template to the item. See also *RenderQueueItem.saveAsTemplate()* and *RenderQueueItem.templates*.

Parameters

`templateName`	A string containing the name of the template to apply.

Returns

Nothing.

30.2.2 RenderQueueItem.duplicate()

`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.

30.2.3 RenderQueueItem.getSetting()

`app.project.renderQueue.item(index).getSetting()`

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Gets a specific Render Queue Item setting.

- Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva
- Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Example

```
// Get current value of render setting's "Proxy Use"
// Key and value strings are English.
var rqItem1_proxyUse = app.project.renderQueue.item(1).getSetting("Proxy Use");

// Get string version of same setting, add "-str" at the end of key string
var rqItem1_proxyUse_str = app.project.renderQueue.item(1).getSetting("Proxy Use-str");
```

30.2.4 RenderQueueItem.getSettings()

`app.project.renderQueue.item(index).getSettings()`

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Gets all settings for a given Render Queue Item.

- Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva
- Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Example

```
// Get object that contains all possible values of all render settings of
// render queue item 1 and convert to JSON format.

var rqItem1_spec_str = app.project.renderQueue.item(1).getSettings(GetSettingsFormat.
↪SPEC);
var rqItem1_spec_str_json = rqItem1_spec_str.toSource();
```

30.2.5 RenderQueueItem.outputModule()

`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.

30.2.6 RenderQueueItem.remove()

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

Description

Removes this item from the render queue.

Parameters

None.

Returns

Nothing.

30.2.7 RenderQueueItem.saveAsTemplate()

`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.

30.2.8 RenderQueueItem.setSetting()

`app.project.renderQueue.item(index).setSetting()`

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Sets a specific setting for a given Render Queue Item.

Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Example

```
// Set value of "Proxy Use" to "Use All Proxies"

app.project.renderQueue.item(1).setSetting("Proxy Use", "Use All Proxies");

// You can use numbers, too.
// The next line does the same as the previous example.

app.project.renderQueue.item(1).setSetting("Proxy Use", 1);
```

30.2.9 RenderQueueItem.setSettings()

`app.project.renderQueue.item(index).setSettings()`

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Sets a multiple settings for a given Render Queue Item.

- Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva
- Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Example

```
// Get an object that contains string version of settable render setting
// values of render queue item 1.
// To get the values in the number format, use
// GetSettingsFormat.NUMBER_SETTABLE as an argument.

var rqItem1_settable_str = app.project.renderQueue.item(1).getSettings(␣
↪GetSettingsFormat.STRING_SETTABLE );

// Set render queue item 2 with values that you got from render
//queue item 1.

app.project.renderQueue.item(2).setSettings( rqItem1_settable_str );

// Set render queue item 3 with values you create.

var my_renderSettings = {
  "Color Depth":        "32 bits per channel",
  "Quality":            "Best",
  "Effects":            "All On",
  "Time Span Duration": "1.0",
  "Time Span Start":    "2.0"
};

app.project.renderQueue.item(2).setSettings( my_renderSettings );
```

CHAPTER

THIRTYONE

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.

> OMCollection is a subclass of *Collection object*. All methods and attributes of Collection are available when working with OMCollection.

CHAPTER

THIRTYTWO

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.

32.1 Attributes

32.1.1 OutputModule.file

`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.

32.1.2 OutputModule.includeSourceXMP

`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.

32.1.3 OutputModule.name

`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.

32.1.4 OutputModule.postRenderAction

`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`

32.1.5 OutputModule.templates

`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.

32.2 Methods

32.2.1 OutputModule.applyTemplate()

`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.

32.2.2 OutputModule.getSetting()

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

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Gets a specific setting for a given Output Module.

- Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva
- Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Example

See the example in *RenderQueueItem.getSetting()* for structure reference.

32.2.3 OutputModule.getSettings()

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

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Gets all settings for a given Output Module.

- Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva
- Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Example

```
// Get object that contains the string version of all current output module setting
// values of output module item 1 from render queue item 1.
// To get the values in the number format, use GetSettingsFormat.NUMBER as an argument.

var omItem1_all_str= app.project.renderQueue.item(1).outputModule(1).getSettings(␣
↪GetSettingsFormat.STRING );

// Convert to JSON format so that it is human-readable.
```

(continues on next page)

(continued from previous page)

```
var omItem1_all_str_json = omItem1_all_str.toSource();

// Get object that contains string version of settable output module setting values
// of output module item 1 from render queue item 1.
// If you want to get the values in the number format, use
// GetSettingsFormat.NUMBER_SETTABLE as an argument.

var omItem1_settable_str = app.project.renderQueue.item(1).outputModule(1).getSettings(␣
↪GetSettingsFormat.STRING_SETTABLE );

// Currently, the format setting in the output module is not settable, but it
// is readable. The next line will tell you the current format of output module
// item 1 from render queue item 1.

var current_format = app.project.renderQueue.item(1).outputModule(1).
↪getSettings(GetSettingsFormat.STRING).Format;

// This line will tell you the output module file info.

var current_omFileTempalte = app.project.renderQueue.item(1).outputModule(1).
↪getSettings(GetSettingsFormat.STRING)["Output File Info"]["File Template"];
```

32.2.4 OutputModule.remove()

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

Description

Removes this OutputModule object from the collection.

Parameters

None.

Returns

Nothing.

32.2.5 OutputModule.saveAsTemplate()

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

Description

Saves this output module as a template and adds it to the te mpl ate s array.

Parameters

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

Returns

Nothing.

32.2.6 OutputModule.setSetting()

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

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

Sets a specific setting for a given Output Module.

- Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva
- Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Example

See the example in *RenderQueueItem.setSetting()* for structure reference.

32.2.7 OutputModule.setSettings()

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

Note: This functionality was added in After Effects 13.0 (CC 2014)

Description

- Depreciated Source: https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva
- Archived version: https://web.archive.org/web/20200622100656/https://blogs.adobe.com/creativecloud/new-changed-after-effects-cc-2014/?segment=dva

Warning: There is a bug that causes OutputModule object to be invalidated after the output module setting is modified, so you need to retrieve the Output Module again after you modify it.

Examples

Get the settings from one item's output module and use them on another:

```
// If you want to get the values in the number format, use
// GetSettingsFormat.NUMBER_SETTABLE as an argument.

var omItem1_settable_str = app.project.renderQueue.item(1).outputModule(1).getSettings(␣
↪GetSettingsFormat.STRING_SETTABLE );
```

(continues on next page)

(continued from previous page)

```
// Set output module item 1 of render queue item 2 with values that you get from
// output module 1 of render queue item 1

app.project.renderQueue.item(2).outputModule(1).setSettings( omItem1_settable_str );
```

Set output module item 1 of render queue item 3 with values that you create:

```
var crop_data = {
  "Crop":        true,
  "Crop Bottom": 0,
  "Crop Left":   0,
  "Crop Right":  8,
  "Crop Top":    10
};

app.project.renderQueue.item(1).outputModule(3).setSettings( crop_data );
```

Route the output file to the user directory:

```
var om1 = app.project.renderQueue.item(1).outputModule(1);
var file_name = File.decode( om1.file.name ); // Name contains special character, space?
var new_dir = new Folder( "~/new_output" );
var new_path = new_dir.fsName;

var new_data = {
  "Output File Info": {
      "Base Path":      new_path,
      "Subfolder Path": "draft",
      "File Name":      file_name
  }
};

om1.setSettings( new_data );
```

In this example, the output file is routed to the user directory, but this time using the full path:

```
var om1 = app.project.renderQueue.item(1).outputModule(1);

// Name contains special character, such as space?
var file_name = File.decode( om1.file.name );
var new_path = "/Users/myAccount/new_output";
var separator = "/";

if ($.os.indexOf("Mac") == -1) {
  new_path = "C:\Users\myAccount\new_output";
  separator = "\\";
}

var new_data = {
  "Output File Info": {
    "Full Flat Path": new_path + separator + file_name
  }
};
```

(continues on next page)

(continued from previous page)

```
om1.setSettings( new_data );
```

CHAPTER

THIRTYTHREE

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 object*. All methods and attributes of FootageSource, in addition to those listed below, are available when working with FileSource.

33.1 Attributes

33.1.1 FileSource.file

```
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.

33.1.2 FileSource.missingFootagePath

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

Description

The path and filename of footage that is missing from this asset. See also *AVItem.footageMissing*.

Type

String; read-only.

33.2 Methods

33.2.1 FileSource.reload()

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

CHAPTER

THIRTYFOUR

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 object*, or the `proxySource` of a *CompItem object* or FootageItem.

> FootageSource is the base class for *SolidSource object*, so FootageSource attributes and methods are available when working with SolidSource objects.

34.1 Attributes

34.1.1 FootageSource.alphaMode

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

Description

Defines how the alpha information in the footage is interpreted. If `hasAlpha` is false, this attribute has no relevant meaning.

Type

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

- `AlphaMode.IGNORE`
- `AlphaMode.STRAIGHT`
- `AlphaMode.PREMULTIPLIED`

34.1.2 FootageSource.conformFrameRate

```
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.

34.1.3 FootageSource.displayFrameRate

```
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.

34.1.4 FootageSource.fieldSeparationType

```
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`

34.1.5 FootageSource.hasAlpha

```
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.

34.1.6 FootageSource.highQualityFieldSeparation

```
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.

34.1.7 FootageSource.invertAlpha

```
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.

34.1.8 FootageSource.isStill

```
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 a duration of 0. Examples of non-still footage are movie files, sound files, sequences, and placeholders of non-zero duration.

Type

Boolean; read-only.

34.1.9 FootageSource.loop

```
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.

34.1.10 FootageSource.nativeFrameRate

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

Description

The native frame rate of the footage.

Type

Floating-point; read-only.

34.1.11 FootageSource.premulColor

```
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.

34.1.12 FootageSource.removePulldown

```
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`

34.2 Methods

34.2.1 FootageSource.guessAlphaMode()

```
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.

34.2.2 FootageSource.guessPulldown()

```
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.

CHAPTER

THIRTYFIVE

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 object*. All methods and attributes of FootageSource are available when working with PlaceholderSource. PlaceholderSource does not define any additional methods or attributes.

CHAPTER

THIRTYSIX

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.

36.1 Attributes

36.1.1 SolidSource.color

```
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.

CHAPTER

THIRTYSEVEN

CHARACTERRANGE OBJECT

```
app.project.item(index).layer(index).text.sourceText.value.
characterRange(characterIndexStart,
[signedCharacterIndexEnd])
```

Note: This functionality was added in After Effects (Beta) 24.2 and is subject to change while it remains in Beta.

Description

The CharacterRange object is an accessor to a character range of the *TextDocument object* instance it was created from.

Unlike the *TextDocument object*, which looks at only the first character when returning character attributes, here the character range can span zero or more characters. As a consequence, two or more characters *may not have the same attribute value* and this mixed state will be signaled by returning `undefined`.

- The *characterStart* attribute is the first character index of the range.
- The *characterEnd* attribue will report the (last + 1) character index of the range, such that (*characterEnd* - *characterStart*) represents the number of characters in the range.

It is acceptable for most attributes for the effective range to be zero - otherwise known as an insertion point.

When accessed, the CharacterRange object will check that *characterStart* and effective *characterEnd* of the range remains valid for the current span of the related *TextDocument object*. This is the same rule as applied when the CharacterRange was created, but because the length of the related *TextDocument object* can change through the addition or removal of characters, the *characterStart* and effective *characterEnd* may no longer be valid. In this situation an exception will be thrown on access, either read or write. The *isRangeValid* attribute will return false if the effective range is no longer valid.

Note that if the *TextDocument object* length changes, the *CharacterRange object* range could become valid again.

Differences from TextDocument

Because CharacterRange is an accessor of *TextDocument object*, most methods and attributes of TextDocument are available when working with CharacterRange. The attributes and methods that are unique to CharacterRange or exhibit unique behaviors are included on this page.

The following attributes and methods are **not** available on instances of CharacterRange:

Attributes	Methods
baselineLocs	*characterRange*
boxText	*paragraphCharacterIndexesAt*
boxTextPos	*paragraphRange*
boxTextSize	
lineOrientation	
paragraphCount	
pointText	

Examples

This increases the font size of the first character in the TextDocument, and set the rest of the characters to fontSize 40.

```
var textDocument = app.project.item(index).layer(index).property("Source Text").value;
var characterRange = textDocument.characterRange(0,1);

characterRange.fontSize = characterRange.fontSize + 5;
textDocument.characterRange(1,-1).fontSize = 40;
```

37.1 Attributes

37.1.1 CharacterRange.characterEnd

`CharacterRange.characterEnd`

Description

The Text layer range calculated character end value.

Throws an exception on access if the effective value would exceed the bounds of the related *TextDocument object*.

Type

Unsigned integer; read-only.

37.1.2 CharacterRange.characterStart

`CharacterRange.characterStart`

Description

The Text layer range calculated character start value.

Throws an exception on access if the effective value would exceed the bounds of the related *TextDocument object*.

Type

Unsigned integer; read-only.

37.1.3 CharacterRange.fillColor

`CharacterRange.fillColor`

Description

The Text layer range CharacterRange attribute 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.

Setting this value will also set `applyFill` to true across the affected characters.

If this attribute has a mixed value for the range of characters, it will be read as `undefined`.

> **Warning:** In contrast to the same attribute on the TextDocument API, we will *not* throw an exception on read if `applyFill` is not true.

Type

Array `[r, g, b]` of floating-point values; read/write.

37.1.4 CharacterRange.isRangeValid

`CharacterRange.isRangeValid`

Description

Returns true if the current range is within the bounds of the related *TextDocument object*, false otherwise.

Type

Boolean; read-only.

37.1.5 CharacterRange.kerning

`CharacterRange.kerning`

Description

The Text layer range character attribute kerning option.

This effectively reports the manual kerning value, and not the calculated kerning value from auto kerning.

- If *autoKernType* in the range is set to `AutoKernType.METRIC_KERN`, `AutoKernType.OPTICAL_KERN`, or is mixed, then this attribute will be returned as `undefined`.
- If *autoKernType* in the range is set to `AutoKernType.NO_AUTO_KERN`, and this attribute has a mixed value, it will be read as `undefined`.

Setting this value will also set `AutoKernType.NO_AUTO_KERN` to true across the affected characters.

Type

Integer value; read/write.

37.1.6 CharacterRange.strokeColor

`CharacterRange.strokeColor`

Description

The Text layer CharacterRange stroke color character property, 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.

If this attribute has a mixed value, it will be read as `undefined`.

Setting this value will also set *applyStroke* to true across the affected characters.

> **Warning:** In contrast to the same attribute on the TextDocument API, we will *not* throw an exception on read if *applyStroke* is not true.

Type

Array [r, g, b] of floating-point values; read/write.

37.1.7 CharacterRange.strokeOverFill

`CharacterRange.strokeOverFill`

Description

The Text layer CharacterRange Stroke Over Fill character property.

Indicates the rendering order for the fill and stroke for characters in the range. When true, the stroke appears over the fill.

If this attribute has a mixed value, it will be read as `undefined`.

> **Warning:**
>
> The Text layer can override per-character attribute setting via the All Strokes First or All Fills First setting on the CharPanel.
>
> The value returned here represents what is applied to the characters, without regard to the possible Text layer override.

Type

Boolean; read/write.

37.1.8 CharacterRange.text

`CharacterRange.text`

Description

The text value for the Text layer range.

On read, the same number of characters as the span of the range will be returned. If the span is zero (an insertion point) it return an empty string.

On write, the characters in the range will be replaced with whatever string value is supplied. If an empty string, then the characters in the range will be effectively deleted.

To insert characters without deleting any existing, call *TextDocument.characterRange()* with the same value for start as end to get an insertion point range.

Type

String; read/write.

37.2 Methods

37.2.1 CharacterRange.toString()

`CharacterRange.toString()`

Description

Returns a string with the parameters used to create the *CharacterRange* instance, e.g. `"CharacterRange(0,-1)"`.

This may be safely called on an instance where *isRangeValid* returns false.

Parameters

None.

Returns

String;

CHAPTER

THIRTYEIGHT

COMPOSEDLINERANGE OBJECT

```
app.project.item(index).layer(index).text.sourceText.value.
composedLineRange(composedLineIndexStart,
[signedComposedLineIndexEnd])
```

Note: This functionality was added in After Effects (Beta) 24.3 and is subject to change while it remains in Beta.

Description

The *ComposedLineRange* object is an accessor to a composed line range of the *TextDocument object* instance it was created from.

Composed lines are initialized in the *TextDocument object* when it is created and remain unchanged while the *TextDocument object* is changed. It is important to note that the *TextDocument object* instance is not re-composed when changes are made to it - that only occurs when the instance is applied back to a *TextLayer object*. So if you delete all the text in the *TextDocument object* instance the number of composed lines will remain constant.

- The *characterStart* attribute will report the first character index of the range.
- The *characterEnd* attribute will report the (last + 1) character index of the range, such that (*characterEnd* - *characterStart*) represents the number of characters in the range.
- A composed line always has some length.

When accessed, the *ComposedLineRange* object will check that effective *characterStart* and effective *characterEnd* of the range remains valid for the current span of the related *TextDocument object*. This is the same rule as applied when the *ComposedLineRange* was created, but because the length of the related *TextDocument object* can change through the addition or removal of characters, the effective *characterStart* and effective *characterEnd* may no longer be valid. In this situation an exception will be thrown on access, either read or write. The property *isRangeValid* will return false if the effective range is no longer valid.

Note that if the *TextDocument object* length changes, the character range could become valid again.

As a convenience, the function *ComposedLineRange.characterRange()* can be invoked which will return a *CharacterRange object* instance initialized from *characterStart* and *characterEnd*. This instance becomes independent of the *ComposedLineRange* instance it came from so subsequent changes to the *ComposedLineRange* limits are not communicated to the *CharacterRange object* instance.

For performance reasons, when accessing multiple attributes it is adviseable to retrieve the *CharacterRange object* once and re-use it rather than create a new one each time.

Examples

This changes the fill color to red of the first composed line in the TextDocument, and set the rest of the lines to color blue.

```
var textDocument = app.project.item(index).layer(index).property("Source Text").value;

var composedLineRange0 = textDocument.composedLineRange(0,1);
var characterRange0 = composedLineRange0.characterRange();
characterRange0.fillColor = [1.0, 0, 0];

textDocument.composedLineRange(1,-1).characterRange().fillColor = [0, 0, 1.0];
```

38.1 Attributes

38.1.1 ComposedLineRange.characterEnd

`ComposedLineRange.characterEnd`

Description

The Text layer range calculated character end value.

Throws an exception on access if the effective value would exceed the bounds of the related *TextDocument object*.

Type

Unsigned integer; read-only.

38.1.2 ComposedLineRange.characterStart

`ComposedLineRange.characterStart`

Description

The Text layer range calculated character start value.

Throws an exception on access if the effective value would exceed the bounds of the related *TextDocument object*.

Type

Unsigned integer; read-only.

38.1.3 ComposedLineRange.isRangeValid

`ComposedLineRange.isRangeValid`

Description

Returns true if the current range is within the bounds of the related *TextDocument object*, false otherwise.

Type

Boolean; read-only.

38.2 Methods

38.2.1 ComposedLineRange.characterRange()

`ComposedLineRange.characterRange()`

Description

Returns a *CharacterRange object* initialized from *characterStart* and *characterEnd*.

Will throw an exception if *isRangeValid* would return false.

The returned instance, once created, is independent of subsequent changes to the *ComposedLineRange* it came from.

Parameters

None.

Returns

CharacterRange object;

38.2.2 ComposedLineRange.toString()

`ComposedLineRange.toString()`

Description

Returns a string with the parameters used to create the *ComposedLineRange* instance, e.g. `"ComposedLineRange(0, -1)"`

This may be safely called on an instance where *isRangeValid* returns false.

Parameters

None.

Returns

String;

CHAPTER

THIRTYNINE

FONT OBJECT

Note: This functionality was added in After Effects 24.0.

Description

The Font object provides information about a specific font, along with the font technology used, helping disambiguate when multiple fonts sharing the same Postscript name are installed on the system.

Most of these APIs simply return information which is contained in the Font data file itself, seek more information there.

39.1 Attributes

39.1.1 FontObject.designAxesData

`app.fonts.allFonts[0][0].designAxesData`

Description

Returns an Array of Objects, containing the design axes data from the font. Each object is composed of the axis `name`, `tag`, `min` value and `max` value.

Note: Will return undefined for non-variable fonts.

Example

This example will select the first returned Font Family Array.

```
// Getting the first available Variable Font on the system
var firstVariableFont = fontsWithDefaultDesignAxes[0];
var axesData = firstVariableFont.designAxesData;

// Getting the first design axis for that Font
var firstAxis = axesData[0];

alert(firstAxis.name+"\n"+firstAxis.tag+"\n"+firstAxis.min+"\n"+firstAxis.max);
```

Type

Array of Objects; read-only.

39.1.2 FontObject.designVector

`app.fonts.fontsWithDefaultDesignAxes[0].designVector`

Description

For Variable fonts will return an ordered array with a length matching the number of design axes defined by the font.

Note: Will return undefined for non-variable fonts.

Type

Array of floating-point values; read-only.

39.1.3 FontObject.familyName

`app.fonts.allFonts[0][0].familyName`

Description

The family name of the font, in the ASCII character set.

Type

String; read-only.

39.1.4 FontObject.familyPrefix

`app.fonts.fontsWithDefaultDesignAxes[0].familyPrefix`

Description

The family prefix of the variable font. For example, the family of the PostScript name “SFPro-Bold” is “SFPro”.

Note: Will return undefined for non-variable fonts.

Type

String; read-only.

39.1.5 FontObject.fontID

`app.fonts.allFonts[0][0].fontID`

Note: This functionality was added in After Effects 24.2.

Description

A unique number assigned to the FontObject instance when it is created, value is greater than or equal to 1. It never changes during the application session but may be different in subsequent launches of the application.

Can be used to compare two FontObject instances to see if they refer to the same underlying native font instance.

FontObjects can be looked up by fontID with *getFontByID* .

Type

Number; read-only.

39.1.6 FontObject.fullName

`app.fonts.allFonts[0][0].fullName`

Description

The full name of the font, in the ASCII character set. Usually composed of the family name and the style name.

Type

String; read-only.

39.1.7 FontObject.hasDesignAxes

`app.fonts.allFonts[0][0].hasDesignAxes`

Description

Returns true if the font is a variable font.

Type

Boolean; read-only.

39.1.8 FontObject.isFromAdobeFonts

`app.fonts.allFonts[0][0].isFromAdobeFonts`

Description

Returns true if the font is from Adobe Fonts.

Type

Boolean; read-only.

39.1.9 FontObject.isSubstitute

`app.fonts.allFonts[0][0].isSubstitute`

Description

Returns true when this font instance represents a font reference which was missing on project open.

Type

Boolean; read-only.

39.1.10 FontObject.location

`app.fonts.allFonts[0][0].location`

Description

The location of the font file on your system.

> **Warning:** Not guaranteed to be returned for all font types; return value may be empty string for some kinds of fonts.

Type

String; read-only.

39.1.11 FontObject.nativeFamilyName

`app.fonts.allFonts[0][0].nativeFamilyName`

Description

The native family name of the font in full 16 bit Unicode. Often different than what is returned by *FontObject.familyName* for non-Latin fonts.

Type

String; read-only.

39.1.12 FontObject.nativeFullName

`app.fonts.allFonts[0][0].nativeFullName`

Description

The native full name of the font in full 16 bit Unicode. Often different than what is returned by *FontObject.fullName* for non-Latin fonts.

Type

String; read-only.

39.1.13 FontObject.nativeStyleName

`app.fonts.allFonts[0][0].nativeStyleName`

Description

The native style name of the font in full 16 bit Unicode. Often different than what is returned by *FontObject.styleName* for non-Latin fonts.

Type

String; read-only.

39.1.14 FontObject.postScriptName

`app.fonts.allFonts[0][0].postScriptName`

Description

The postscript name of the font.

Type

String; read-only.

39.1.15 FontObject.styleName

`app.fonts.allFonts[0][0].styleName`

Description

The style name of the font, in the ASCII character set.

Type

String; read-only.

39.1.16 FontObject.technology

`app.fonts.allFonts[0][0].technology`

Description

The technology used by the font.

Type

An `CTFontTechnology` enumerated value; read-only. One of:

- `CTFontTechnology.CT_TYPE1_FONT`
- `CTFontTechnology.CT_TRUETYPE_FONT`
- `CTFontTechnology.CT_CID_FONT`
- `CTFontTechnology.CT_BITMAP_FONT`

- CTFontTechnology.CT_ATC_FONT
- CTFontTechnology.CT_TYPE3_FONT
- CTFontTechnology.CT_SVG_FONT
- CTFontTechnology.CT_ANYTECHNOLOGY

39.1.17 FontObject.type

app.fonts.allFonts[0][0].type

Description

The internal type of the font.

Type

An CTFontType enumerated value; read-only. One of:

- CTFontType.CT_TYPE1_FONTTYPE
- CTFontType.CT_TRUETYPE_FONTTYPE
- CTFontType.CT_CID_FONTTYPE
- CTFontType.CT_ATC_FONTTYPE
- CTFontType.CT_BITMAP_FONTTYPE
- CTFontType.CT_OPENTYPE_CFF_FONTTYPE
- CTFontType.CT_OPENTYPE_CID_FONTTYPE
- CTFontType.CT_OPENTYPE_TT_FONTTYPE
- CTFontType.CT_TYPE3_FONTTYPE
- CTFontType.CT_SVG_FONTTYPE

39.1.18 FontObject.version

app.fonts.allFonts[0][0].version

Description

The version number of the font.

Type

String; read-only.

39.1.19 FontObject.writingScripts

`app.fonts.allFonts[0][0].writingScripts`

Description

The supported character sets of the font.

Type

An array of `CTScript` enumerated value; read-only. One or more of:

- `CTScript.CT_ROMAN_SCRIPT`
- `CTScript.CT_JAPANESE_SCRIPT`
- `CTScript.CT_TRADITIONALCHINESE_SCRIPT`
- `CTScript.CT_KOREAN_SCRIPT`
- `CTScript.CT_ARABIC_SCRIPT`
- `CTScript.CT_HEBREW_SCRIPT`
- `CTScript.CT_GREEK_SCRIPT`
- `CTScript.CT_CYRILLIC_SCRIPT`
- `CTScript.CT_RIGHTLEFT_SCRIPT`
- `CTScript.CT_DEVANAGARI_SCRIPT`
- `CTScript.CT_GURMUKHI_SCRIPT`
- `CTScript.CT_GUJARATI_SCRIPT`
- `CTScript.CT_ORIYA_SCRIPT`
- `CTScript.CT_BENGALI_SCRIPT`
- `CTScript.CT_TAMIL_SCRIPT`
- `CTScript.CT_TELUGU_SCRIPT`
- `CTScript.CT_KANNADA_SCRIPT`
- `CTScript.CT_MALAYALAM_SCRIPT`
- `CTScript.CT_SINHALESE_SCRIPT`
- `CTScript.CT_BURMESE_SCRIPT`
- `CTScript.CT_KHMER_SCRIPT`
- `CTScript.CT_THAI_SCRIPT`
- `CTScript.CT_LAOTIAN_SCRIPT`
- `CTScript.CT_GEORGIAN_SCRIPT`
- `CTScript.CT_ARMENIAN_SCRIPT`
- `CTScript.CT_SIMPLIFIEDCHINESE_SCRIPT`
- `CTScript.CT_TIBETAN_SCRIPT`
- `CTScript.CT_MONGOLIAN_SCRIPT`
- `CTScript.CT_GEEZ_SCRIPT`
- `CTScript.CT_EASTEUROPEANROMAN_SCRIPT`

- CTScript.CT_VIETNAMESE_SCRIPT
- CTScript.CT_EXTENDEDARABIC_SCRIPT
- CTScript.CT_KLINGON_SCRIPT
- CTScript.CT_EMOJI_SCRIPT
- CTScript.CT_ROHINGYA_SCRIPT
- CTScript.CT_JAVANESE_SCRIPT
- CTScript.CT_SUNDANESE_SCRIPT
- CTScript.CT_LONTARA_SCRIPT
- CTScript.CT_SYRIAC_SCRIPT
- CTScript.CT_TAITHAM_SCRIPT
- CTScript.CT_BUGINESE_SCRIPT
- CTScript.CT_BALINESE_SCRIPT
- CTScript.CT_CHEROKEE_SCRIPT
- CTScript.CT_MANDAIC_SCRIPT
- CTScript.CT_VAI_SCRIPT
- CTScript.CT_THAANA_SCRIPT
- CTScript.CT_BRAVANESE_SCRIPT
- CTScript.CT_BRAHMI_SCRIPT
- CTScript.CT_CARIAN_SCRIPT
- CTScript.CT_CYPRIOT_SCRIPT
- CTScript.CT_EGYPTIAN_SCRIPT
- CTScript.CT_IMPERIALARAMAIC_SCRIPT
- CTScript.CT_PAHLAVI_SCRIPT
- CTScript.CT_PARTHIAN_SCRIPT
- CTScript.CT_KHAROSHTHI_SCRIPT
- CTScript.CT_LYCIAN_SCRIPT
- CTScript.CT_LYDIAN_SCRIPT
- CTScript.CT_PHOENICIAN_SCRIPT
- CTScript.CT_PERSIAN_SCRIPT
- CTScript.CT_SHAVIAN_SCRIPT
- CTScript.CT_SUMAKKCUNEIFORM_SCRIPT
- CTScript.CT_UGARITIC_SCRIPT
- CTScript.CT_GLAGOLITIC_SCRIPT
- CTScript.CT_GOTHIC_SCRIPT
- CTScript.CT_OGHAM_SCRIPT
- CTScript.CT_OLDITALIC_SCRIPT

- CTScript.CT_ORKHON_SCRIPT
- CTScript.CT_RUNIC_SCRIPT
- CTScript.CT_MEROITICCURSIVE_SCRIPT
- CTScript.CT_COPTIC_SCRIPT
- CTScript.CT_OLCHIKI_SCRIPT
- CTScript.CT_SORASOMPENG_SCRIPT
- CTScript.CT_OLDHANGUL_SCRIPT
- CTScript.CT_LISU_SCRIPT
- CTScript.CT_NKO_SCRIPT
- CTScript.CT_ADLAM_SCRIPT
- CTScript.CT_BAMUM_SCRIPT
- CTScript.CT_BASSAVAH_SCRIPT
- CTScript.CT_NEWA_SCRIPT
- CTScript.CT_NEWTAILU_SCRIPT
- CTScript.CT_SCRIPT
- CTScript.CT_OSAGE_SCRIPT
- CTScript.CT_UCAS_SCRIPT
- CTScript.CT_TIFINAGH_SCRIPT
- CTScript.CT_KAYAHLI_SCRIPT
- CTScript.CT_LAO_SCRIPT
- CTScript.CT_TAILE_SCRIPT
- CTScript.CT_TAIVIET_SCRIPT
- CTScript.CT_DONTKNOW_SCRIPT

39.2 Methods

39.2.1 FontObject.hasSameDict()

`app.fonts.fontsWithDefaultDesignAxes[0].hasSameDict(fontObject)`

Description

This function will true if the *Font object* passed as an argument shares the same variable font dictionnary as the *Font object* the function is called on.

Note: Can only return true when called on a variable *Font object* with the argument also being a *Font object* of a variable font.

Parameters

fontObject	A *Font object*

Returns

A Boolean.

39.2.2 FontObject.postScriptNameForDesignVector()

`app.fonts.fontsWithDefaultDesignAxes[0].postScriptNameForDesignVector([...vectorValues])`

Description

This function will return the postscript name of the variable font for the specific design vectors passed as the argument.

Parameters

vectorValues	An array of float values that matches the length of *FontObject.designVector* for the given variable font.

Returns

A String.

CHAPTER

FORTY

FONTS OBJECT

`app.fonts`

Note: This functionality was added in After Effects 24.0.

Description

The Fonts objects provides information about the current font ecosystem on your device.

After Effects maintains an internal font proxy to a real font which it has enumerated in the font ecosystem. As the fonts in the font ecosystem are added and removed these internal font proxies are kept in sync as well by being added and removed.

The properties we report via the proxy *Font object* are the data that is available to us from the font files themselves, which of course will vary according to technology and type of font. It is not possible here to describe all the possible interesting variations and troubles that this causes us and in general it is advisable to be careful with assuming that the behavior and properties for one font type or technology are common to all other font types and technology - the answer as always is "it depends".

A *Font object* is a soft reference to one of these internal font proxies and as a consequence is not sufficient to keep the internal font proxy alive. As a result if the internal font proxy is removed, the referencing *Font object* will throw an invalid exception for any property reference.

On project open, and a few other situations, it may come to pass that the font which is being referenced in the persisted data cannot be found in the current font ecosystem. In these situations an internal font proxy will be created which will contain the desired properties, such as PostScript name, and will return true for *isSubstitute*. There will be an underlying real font which will be selected to support this internal font proxy, but we do not reveal what it is and there is no way to influence this selection.

Continuing the open process with created substitute fonts, an attempt will be made to sync matching fonts from Creative Cloud Adobe Fonts. This is an asynchronous activity and the project will usually finish opening and be ready for use before any fonts are brought down from Adobe Fonts. Depending on how many fonts are being synced, they may be installed at different times. There is no way to disable this attempt.

After any change to the font ecosystem from installing new real fonts, the outstanding list of substitute fonts will be evaluated to see if there now exists a real font which is a valid replacement for it - currently only requiring the PostScript name to match - and if one is found automatically all the references in the project to the substitute will be replaced with the newly installed font.

40.1 Attributes

40.1.1 FontsObject.allFonts

`app.fonts.allFonts`

Description

The list of all the fonts currently available on your system.

They are grouped into what is named a family group which are Arrays of *Font object*.

The Family Name of the group is simply the *familyName* of any of the *Font objects* in the group.

The Family Name in one font group is not guaranteed to have unique name compared to different font groups - the grouping is determined by a number of factors including the returned value of *FontObject.technology* and *FontObject.writingScripts*.

In addition, it is perfectly acceptable to have multiple fonts with the same PostScript name, though only one will have the same (PostScript name, Technology, Primary Writing Script) tuple. In the case of true duplicates, it is undefined which will be returned and which will be suppressed.

The family groups and *Font objects* in the group are sorted according to the setting in the Character Panel dropdown "Show Font Names in English". If set to true, the *familyName* and *styleName* property is used, otherwise the *nativeFamilyName* and *nativeStyleName* property is used.

Font object for which *isSubstitute* returns true are always sorted to the end as individual family groups.

Type

Array of Arrays of *Font objects*; read-only.

Example

This example will select the first returned Font Family Array.

```
// Getting the first available Font Family Group on the system
var firstFontGroup = app.fonts.allFonts[0];

// Getting the first Style for that Font Family
var firstFontFamilyName = firstFontGroup[0].familyName;
var firstFamilyStyle = firstFontGroup[0].styleName;

alert(firstFontFamilyName+" "+firstFamilyStyle);
```

40.1.2 FontsObject.favoriteFontFamilyList

`app.fonts.favoriteFontFamilyList`

Note: This functionality was added in After Effects (Beta) 24.4 and subject to change while it remains in Beta.

Description

Provides access to the Favorites list presented in the Character panel and Properties panel. To set the Favorites simply provide an (unsorted) array of strings based on the *familyName*. To clear the list simply assign an empty Array.

Type

Array of Strings; read/write.

40.1.3 FontsObject.fontsDuplicateByPostScriptName

`app.fonts.fontsDuplicateByPostScriptName`

Note: This functionality was added in After Effects (Beta) 24.4 and subject to change while it remains in Beta.

Description

It is perfectly legal and common for more than one *Font object* to return the same value for *postScriptName* but as this can sometimes lead to confusion about what *Font object* will actually be used when using *TextDocument.font* or the `.font` attribute of a *CharacterRange object*, this property exists to both reveal what duplicates exist and also their relative order.

This an Array in which each element is an Array of *Font objects*, where the 0th element *Font object* is considered the primary *Font object* for the given PostScript name.

Type

Array of Arrays of *Font object*; read-only.

40.1.4 FontsObject.fontServerRevision

`app.fonts.fontServerRevision`

Note: This functionality was added in After Effects 24.2.

Description

Returns an unsigned number representing the current revision of the font environment.

The revision is advanced when anything happens to the font environment which would change the contents, properties, or order of *Font objects* returned from a call to *FontsObject.allFonts*.

Among these are: installing or removing fonts in the font environment, opening or closing a project with substituted fonts, causing a custom Variable font instance to be created, and changing the setting in the Character Panel dropdown "Show Font Names in English".

Type

Number; read-only.

Example

```
var fsRev = app.fonts.fontServerRevision;
alert(fsRev);
```

40.1.5 FontsObject.fontsWithDefaultDesignAxes

`app.fonts.fontsWithDefaultDesignAxes`

Description

Returns an array of variable *Font objects*, each using a unique font dictionary and with default values for their design axes. This API is a convenient way to quickly filter for a unique instance of each installed variable font.

Type

Array of *Font objects*; read-only.

Example

```
var variableFontList = app.fonts.fontsWithDefaultDesignAxes;
alert(variableFontList.length);
```

40.1.6 FontsObject.freezeSyncSubstitutedFonts

`app.fonts.freezeSyncSubstitutedFonts`

Note: This functionality was added in After Effects (Beta) 24.4 and subject to change while it remains in Beta.

Description

When a Project is opened and one or more fonts are not found in the local font environment, a *sync* process is initiated with Adobe Fonts to see if one or more Fonts could be activated and installed.

By default this happens automatically—this property will disable it from happening.

Warning: The rules for deciding if Adobe Fonts has a matching font is entirely based on the PostScript name. With some Variable Fonts, due to ambiguity about which font has which named instance, it is possible that more than one face (Regular/Italic) may be installed during an activation. Whether the installed font is a valid replacement is controlled by the *FontsObject.substitutedFontReplacementMatchPolicy*.

Type

Boolean; read/write. One of:

- `false` is the default—sync from Adobe Fonts may be attempted.
- `true` means that no sync or install will be attempted.

40.1.7 FontsObject.missingOrSubstitutedFonts

`app.fonts.missingOrSubstitutedFonts`

Description

The list of all the missing or substituted fonts of the current Project.

Note: A substituted font is a font that was already missing when the project is opened. A missing font is a font that went missing (font being uninstalled, for example) *while* the project was open.

Type

Array of *Font objects*; read-only.

40.1.8 FontsObject.mruFontFamilyList

`app.fonts.mruFontFamilyList`

Note: This functionality was added in After Effects (Beta) 24.4 and subject to change while it remains in Beta.

Description

Provides access to the Most Recently Used (MRU) list presented in the Character panel and Properties panel. To set the MRU simply provide an (unsorted) array of strings based on the *familyName*. To clear the list simply assign an empty Array.

Type

Array of Strings; read/write.

40.1.9 FontsObject.substitutedFontReplacementMatchPolicy

`app.fonts.substitutedFontReplacementMatchPolicy`

Note: This functionality was added in After Effects (Beta) 24.4 and subject to change while it remains in Beta.

Description

Controls the rules which are used to determine which fonts are considered matching for automatic replacement for a substituted *Font object*.

Type

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

- `SubstitutedFontReplacementMatchPolicy.POSTSCRIPT_NAME` is the default; any *Font object* which has the same PostScript name is a valid candidate for replacement of a substituted *Font object*.
- `SubstitutedFontReplacementMatchPolicy.CTFI_EQUAL` requires that the following properties of substituted *Font object* must match to be considered a valid candidate:

  - *postScriptName*
  - *technology*
  - *writingScripts* (primary)
  - *designVector*

- `SubstitutedFontReplacementMatchPolicy.DISABLED` means that no *Font object* is an acceptable replacement for a the substituted *Font object*.

40.2 Methods

40.2.1 FontsObject.getFontByID()

`app.fonts.getFontByID(fontID)`

Note: This functionality was added in After Effects 24.2.

Description

This function will return an instance of *Font object* based on the ID of a previously found font.

If no matching font is found, it will return undefined. This can occur with an unknown ID or if the original font has been removed from the font environment.

```
var font1 = app.fonts.allFonts[0][0];
var font2 = app.fonts.getFontByID(font1.fontID);
alert(font1.fontID == font2.fontID);
```

Parameters

fontID	A number containing the ID of the font.

Returns

Font object, or undefined if no font by that ID.

40.2.2 FontsObject.getFontsByFamilyNameAndStyleName()

`app.fonts.getFontsByFamilyNameAndStyleName(familyName, styleName)`

Description

This function will return an array of *Font object* based on the Family Name and Style Name of a font. If no suitable font is found, it will return an empty Array.

Note: The returned array length can be more than 1 if you have multiple copies of a same font.

```
var fontList = app.fonts.getFontsByFamilyNameAndStyleName("Abolition", "Regular")
alert(fontList.length);
```

Parameters

FamilyName	A string containing the Family Name of the font.
StyleName	A string containing the Style Name of the font.

Returns

Array of *Font objects*; read-only.

40.2.3 FontsObject.getFontsByPostScriptName()

`app.fonts.getFontsByPostScriptName(postscriptName)`

Description

This function will return an array of *Font objects* based on the PostScript name of previously found Fonts.

It is perfectly valid to have multiple *Font objects* which share the same PostScript name, the order of these is determined by the order in which they were enumerated in the font environment. The entry at `[0]` will be used when setting the *TextDocument.fontObject* property.

In addition, there is a special property of this API with regards to Variable fonts. If no *Font object* matching the requested PostScript exists, but we find that there exist a variable font which matches the requested PostScript name prefix, then this Variable font instance will be requested to create a matching *Font object*. This is the only way that we will return an instance which did not exist prior to invoking this method.

If no matching font is found, it will return an empty Array.

```
var fontList = app.fonts.getFontsByPostScriptName("Abolition")
alert(fontList.length);
```

Parameters

postscriptName	A string containing the PostScript Name of the font.

Returns

Array of *Font objects*; read-only.

40.2.4 FontsObject.pollForAndPushNonSystemFontFoldersChanges()

`app.fonts.pollForAndPushNonSystemFontFoldersChanges()`

Note: This functionality was added in After Effects (Beta) 24.4 and subject to change while it remains in Beta.

Description

The addition and removal of font files in what is considered the *system font folders* is recognized and handled automatically without user intervention to update the font environment. Non-system font folders are not automatically handled and so additions and removal of font files in these folders are not recognized until the After Effects is restarted.

This function will trigger a check against the known non-system font folders, and if it is recognized that there has been a change, an asynchronous update to the font environment will be scheduled to process this change.

The non-system font folders After Effects knows about are here:

```
Windows: <systemDrive>:\Program Files\Common Files\Adobe\Fonts

Mac: /Library/Application Support/Adobe/Fonts
```

Returns

Boolean; One of:

- `false` if no changes to the font environment are known.
- `true` if a change in the font environment has been detected and an asynchronous update scheduled to deal with it. This state will be cleared once the update has been processed, at which time *FontsObject.fontServerRevision* will return an incremented value.

CHAPTER

FORTYONE

PARAGRAPHRANGE OBJECT

```
app.project.item(index).layer(index).text.sourceText.value.
paragraphRange(paragraphIndexStart,
[signedParagraphIndexEnd])
```

Note: This functionality was added in After Effects (Beta) 24.2 and is subject to change while it remains in Beta.

Description

The ParagraphRange object is an accessor to a paragraph range of the *TextDocument object* instance it was created from.

- The *characterStart* attribute will report the first character index of the range.
- The *characterEnd* attribute will report the (last + 1) character index of the range, such that (*characterEnd* - *characterStart*) represents the number of characters in the range.
- The only time these two properties will equal will on an empty last paragraph of the *TextDocument object*.

When accessed, the ParagraphRange object will check that effective *characterStart* and effective *characterEnd* of the range remains valid for the current span of the related *TextDocument object*. This is the same rule as applied when the ParagraphRange was created, but because the length of the related *TextDocument object* can change through the addition or removal of characters, the effective *characterStart* and effective *characterEnd* may no longer be valid. In this situation an exception will be thrown on access, either read or write. The *isRangeValid* attribute will return false if the effective range is no longer valid.

Note that if the *TextDocument object* length changes, the character range could become valid again.

As a convenience, the function *ParagraphRange.characterRange()* can be invoked which will return a *CharacterRange object* instance initialized from *characterStart* and *characterEnd*. This instance becomes independent of the ParagraphRange instance it came from so subsequent changes to the ParagraphRange limits are not communicated to the *CharacterRange object* instance.

For performance reasons, when accessing multiple attributes it is adviseable to retrieve the *CharacterRange object* once and re-use it rather than create a new one each time.

Examples

This increases the font size of the first paragraph in the TextDocument, and set the rest of the paragraphs to fontSize 40.

```
var textDocument = app.project.item(index).layer(index).property("Source Text").value;

var paragraphRange0 = textDocument.paragraphRange(0,1);
```

(continues on next page)

(continued from previous page)

```
var characterRange0 = paragraphRange0.characterRange();
characterRange0.fontSize = characterRange0.fontSize + 5;

textDocument.paragraphRange(1,-1).characterRange().fontSize = 40;
```

41.1 Attributes

41.1.1 ParagraphRange.characterEnd

`ParagraphRange.characterEnd`

Description

The Text layer range calculated character end value.

Throws an exception on access if the effective value would exceed the bounds of the related *TextDocument object*.

Type

Unsigned integer; read-only.

41.1.2 ParagraphRange.characterStart

`ParagraphRange.characterStart`

Description

The Text layer range calculated character start value.

Throws an exception on access if the effective value would exceed the bounds of the related *TextDocument object*.

Type

Unsigned integer; read-only.

41.1.3 ParagraphRange.isRangeValid

`ParagraphRange.isRangeValid`

Description

Returns true if the current range is within the bounds of the related *TextDocument object*, false otherwise.

Type

Boolean; read-only.

41.2 Methods

41.2.1 ParagraphRange.characterRange()

`ParagraphRange.characterRange()`

Description

Returns a *CharacterRange object* initialized from *characterStart* and *characterEnd*.

Will throw an exception if *isRangeValid* would return false.

The returned instance, once created, is independent of subsequent changes to the ParagraphRange it came from.

Parameters

None.

Returns

CharacterRange object;

41.2.2 ParagraphRange.toString()

`ParagraphRange.toString()`

Description

Returns a string with the parameters used to create the *ParagraphRange* instance, e.g. `"ParagraphRange(0,-1)"`

This may be safely called on an instance where *isRangeValid* returns false.

Parameters

None.

Returns

String;

CHAPTER

FORTYTWO

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("HappyCake");
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, newTextDocument("Happy"));
textProp.setValueAtTime(.33, newTextDocument("cake"));
textProp.setValueAtTime(.66, newTextDocument("is"));
textProp.setValueAtTime(1, newTextDocument("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 = "Times New Roman PSMT";
textDocument.strokeOverFill = true;
textDocument.applyStroke = true;
textDocument.applyFill = true;
textDocument.text = myString;
textDocument.justification = ParagraphJustification.CENTER_JUSTIFY;
textDocument.tracking = 50;
textProp.setValue(textDocument);
```

42.1 Attributes

42.1.1 TextDocument.allCaps

`textDocument.allCaps`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

True if a Text layer has All Caps enabled; otherwise false. To set this value, use *fontCapsOption* added in After Effects 24.0.

Warning: This value only reflects the first character in the Text layer.

Type

Boolean; read-only.

42.1.2 TextDocument.applyFill

`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.

42.1.3 TextDocument.applyStroke

`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.

42.1.4 TextDocument.autoHyphenate

`textDocument.autoHyphenate`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's auto hyphenate paragraph option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.5 TextDocument.autoLeading

`textDocument.autoLeading`

Description

The Text layer's auto leading character option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.6 TextDocument.autoKernType

`textDocument.autoKernType`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's auto kern type option.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

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

- `AutoKernType.NO_AUTO_KERN`
- `AutoKernType.METRIC_KERN`
- `AutoKernType.OPTICAL_KERN`

42.1.7 TextDocument.baselineDirection

`textDocument.baselineDirection`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's baseline direction option. This is significant for Japanese language in vertical texts. "BASELINE_VERTICAL_CROSS_STREAM" is also know as Tate-Chu-Yoko.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

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

- `BaselineDirection.BASELINE_WITH_STREAM`
- `BaselineDirection.BASELINE_VERTICAL_ROTATED`
- `BaselineDirection.BASELINE_VERTICAL_CROSS_STREAM`

42.1.8 TextDocument.baselineLocs

`textDocument.baselineLocs`

Note: This functionality was added in After Effects 13.6 (CC 2015)

Description

The baseline (x,y) locations for a Text layer. Line wraps in a paragraph text box are treated as multiple lines.

Note: If a line has no characters, the x and y values for start and end will be the maximum float value (3.402823466e+38F).

Type

Array of floating-point values in the form of

```
[
  line0.start_x,
  line0.start_y,
  line0.end_x,
  line0.end_y,
  line1.start_x,
  line1.start_y,
  line1.end_x,
  line1.end_y,
  ...
  lineN-1.start_x,
  lineN-1.start_y,
  lineN-1.end_x,
  lineN-1.end_y
]
```

42.1.9 TextDocument.baselineShift

`textDocument.baselineShift`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

This Text layer's baseline shift in pixels.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Floating-point value; read-write.

42.1.10 TextDocument.boxAutoFitPolicy

`textDocument.boxAutoFitPolicy`

Note: This functionality was added in After Effects (Beta) 24.3 and subject to change while it remains in Beta.

Description

Enables the automated change of the box height to fit the text content in the box. The box only grows down.

Defaults to `BoxAutoFitPolicy.NONE`.

Will be disabled if *TextDocument.boxVerticalAlignment* is anything other than `boxVerticalAlignment.TOP`.

Type

A `BoxAutoFitPolicy` enumerated value; read-write. One of:

- `BoxAutoFitPolicy.NONE`
- `BoxAutoFitPolicy.HEIGHT_CURSOR`
- `BoxAutoFitPolicy.HEIGHT_PRECISE_BOUNDS`
- `BoxAutoFitPolicy.HEIGHT_BASELINE`

42.1.11 TextDocument.boxFirstBaselineAlignment

`textDocument.boxFirstBaselineAlignment`

Note: This functionality was added in After Effects (Beta) 24.3 and subject to change while it remains in Beta.

Description

Controls the position of the first line of composed text relative to the top of the box.

Disabled if *TextDocument.boxFirstBaselineAlignmentMinimum* is anything other than zero.

Defaults to `BoxFirstBaselineAlignment.ASCENT`.

Type

A `BoxFirstBaselineAlignment` enumerated value; read-write. One of:

- `BoxFirstBaselineAlignment.ASCENT`
- `BoxFirstBaselineAlignment.CAP_HEIGHT`
- `BoxFirstBaselineAlignment.EM_BOX`
- `BoxFirstBaselineAlignment.LEADING`
- `BoxFirstBaselineAlignment.LEGACY_METRIC`
- `BoxFirstBaselineAlignment.MINIMUM_VALUE_ASIAN`
- `BoxFirstBaselineAlignment.MINIMUM_VALUE_ROMAN`
- `BoxFirstBaselineAlignment.TYPO_ASCENT`
- `BoxFirstBaselineAlignment.X_HEIGHT`

42.1.12 TextDocument.boxFirstBaselineAlignmentMinimum

`textDocument.boxFirstBaselineAlignmentMinimum`

Note: This functionality was added in After Effects (Beta) 24.3 and subject to change while it remains in Beta.

Description

Manually controls the position of the first line of composed text relative to the top of the box.

A value set here other than zero will override the effect of the *TextDocument.boxFirstBaselineAlignment* value.

Defaults to zero.

Type

Floating-point value; read/write.

42.1.13 TextDocument.boxInsetSpacing

`textDocument.boxInsetSpacing`

Note: This functionality was added in After Effects (Beta) 24.3 and subject to change while it remains in Beta.

Description

Controls the inner space between the box bounds and where the composable text box begins. The same value is applied to all four sides of the box.

Defaults to zero.

Type

Floating-point value; read/write.

42.1.14 TextDocument.boxOverflow

`textDocument.boxOverflow`

Note: This functionality was added in After Effects (Beta) 24.3 and subject to change while it remains in Beta.

Description

Returns true if some part of the text did not compose into the box.

Type

Boolean; read-only.

42.1.15 TextDocument.boxText

`textDocument.boxText`

Description

True if a Text layer is a layer of paragraph (bounded) text; otherwise false.

Type

Boolean; read-only.

42.1.16 TextDocument.boxTextPos

`textDocument.boxTextPos`

Note: This functionality was added in After Effects 13.2 (CC 2014.2) As of After Effects 14 (CC2017), it seems this is also writeable.

Description

The layer coordinates from a paragraph (box) Text layer's anchor point as a [width, height] array of pixel dimensions.

Warning: Throws an exception if *boxText* does not return true for the Text layer.

Type

Array of ([X,Y]) position coordinates; read/write.

Example

```
// For a paragraph Text layer returns [x, y] position from layer anchor point in layer␣
↪coordinates.
// e.g. approximately [0, -25] with default character panel settings.
var boxTextLayerPos = myTextLayer.sourceText.value.boxTextPos;
```

42.1.17 TextDocument.boxTextSize

`textDocument.boxTextSize`

Description

The size of a paragraph (box) Text layer as a [width, height] array of pixel dimensions.

Warning: Throws an exception if *boxText* does not return true for the Text layer.

Type

Array of two integers (minimum value of 1); read/write.

42.1.18 TextDocument.boxVerticalAlignment

`textDocument.boxVerticalAlignment`

Note: This functionality was added in After Effects (Beta) 24.3 and subject to change while it remains in Beta.

Description

Enables the automated vertical alignment of the composed text in the box.

Defaults to `BoxVerticalAlignment.TOP`

Type

A `BoxVerticalAlignment` enumerated value; read-write. One of:

- `BoxVerticalAlignment.TOP`
- `BoxVerticalAlignment.CENTER`
- `BoxVerticalAlignment.BOTTOM`
- `BoxVerticalAlignment.JUSTIFY`

42.1.19 TextDocument.composedLineCount

`textDocument.composedLineCount`

Description

Returns the number of composed lines in the Text layer, may be zero if all text is overset.

The *TextDocument object* instance is initialized from the composed state and subsequent changes to the *TextDocument object* instance does not cause recomposition.

Even if you remove all the text from the *TextDocument object* instance, the value returned here remains unchanged.

Type

Number; read-only.

42.1.20 TextDocument.composerEngine

`textDocument.composerEngine`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph composer engine option. By default new Text layers will use the `ComposerEngine.UNIVERSAL_TYPE_ENGINE`; the other enum value will only be encountered in projects created before the Universal Type Engine engine (formerly known as the South Asian and Middle Eastern engine) became the default in After Effects 22.1.1.

If this attribute has a mixed value, it will be read as `undefined`.

This attrribute is read-write, but an exception will be thrown if any enum value other than `ComposerEngine.UNIVERSAL_TYPE_ENGINE` is written.

In effect, you can change an older document from `ComposerEngine.LATIN_CJK_ENGINE` to `ComposerEngine.UNIVERSAL_TYPE_ENGINE`, but not the reverse.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

A `ComposerEngine` enumerated value; read-write. One of:

- `ComposerEngine.LATIN_CJK_ENGINE`
- `ComposerEngine.UNIVERSAL_TYPE_ENGINE`

42.1.21 TextDocument.digitSet

`textDocument.digitSet`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's digit set option.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

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

- `DigitSet.DEFAULT_DIGITS`
- `DigitSet.ARABIC_DIGITS`
- `DigitSet.HINDI_DIGITS`
- `DigitSet.FARSI_DIGITS`
- `DigitSet.ARABIC_DIGITS_RTL`

42.1.22 TextDocument.direction

`textDocument.direction`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph direction option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

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

- `ParagraphDirection.DIRECTION_LEFT_TO_RIGHT`
- `ParagraphDirection.DIRECTION_RIGHT_TO_LEFT`

42.1.23 TextDocument.endIndent

`textDocument.endIndent`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph end indent option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Floating-point value; read/write.

42.1.24 TextDocument.everyLineComposer

`textDocument.everyLineComposer`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's Every-Line Composer paragraph option. If set to false, the TextDocument will use the Single-Line Composer.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.25 TextDocument.fauxBold

`textDocument.fauxBold`

Note:

The read functionality was added in After Effects 13.2 (CC 2014.2).
The write functionality was added in After Effects 24.0.

Description

True if a Text layer has faux bold enabled; otherwise false.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Boolean; read/write.

Example

```
var isFauxBold = myTextLayer.sourceText.value.fauxBold;
```

42.1.26 TextDocument.fauxItalic

`textDocument.fauxItalic`

Note:

The read functionality was added in After Effects 13.2 (CC 2014.2).
The write functionality was added in After Effects 24.0.

Description

True if a Text layer has faux italic enabled; otherwise false.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.27 TextDocument.fillColor

`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.

Throws an exception on read if *applyFill* is not true.

Setting this value will also set *applyFill* to true across the affected characters.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Array `[r, g, b]` of floating-point values; read/write.

42.1.28 TextDocument.firstLineIndent

`textDocument.firstLineIndent`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph first line indent option.

If this attribute has a mixed value, it will be read as `undefined`.

> **Warning:** This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Floating-point value; read/write.

42.1.29 TextDocument.font

`textDocument.font`

Description

The Text layer's font specified by its PostScript name.

On write, there are very few resrictions on what can be supplied - if the underlying font management system does not have a matching *Font object* instance matching the supplied PostScript name a substitute instance will be created. The Font instance returned in the case of duplicate PostScript names will be the 0th element of the array returned from *FontsObject.getFontsByPostScriptName()*.

You should use the *Font object* attribute for precise control.

> **Warning:** This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

String; read/write.

42.1.30 TextDocument.fontBaselineOption

`textDocument.fontBaselineOption`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's font baseline option. This is for setting a textDocument to superscript or subscript.

> **Warning:** This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

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

- `FontBaselineOption.FONT_NORMAL_BASELINE`
- `FontBaselineOption.FONT_FAUXED_SUPERSCRIPT`
- `FontBaselineOption.FONT_FAUXED_SUBSCRIPT`

42.1.31 TextDocument.fontCapsOption

`textDocument.fontCapsOption`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's font caps option.

> **Warning:** This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

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

- `FontCapsOption.FONT_NORMAL_CAPS`
- `FontCapsOption.FONT_SMALL_CAPS`
- `FontCapsOption.FONT_ALL_CAPS`
- `FontCapsOption.FONT_ALL_SMALL_CAPS`

42.1.32 TextDocument.fontFamily

`textDocument.fontFamily`

Note: This functionality was added in After Effects 13.1 (CC 2014.1)

Description

String with with the name of the font family.

> **Warning:** This value only reflects the first character in the Text layer.

Type

String; read-only.

42.1.33 TextDocument.fontLocation

`textDocument.fontLocation`

Note: This functionality was added in After Effects 13.1 (CC 2014.1)

Description

Path of font file, providing its location on disk.

Warning: Not guaranteed to be returned for all font types; return value may be empty string for some kinds of fonts.

Warning: This value only reflects the first character in the Text layer.

Type

String; read-only.

42.1.34 TextDocument.fontObject

`textDocument.fontObject`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's *Font object* specified by its PostScript name.

Warning: This value only reflects the first character in the Text layer.

Type

Font object; read/write.

42.1.35 TextDocument.fontSize

`textDocument.fontSize`

Description

The Text layer's font size in pixels.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Floating-point value (0.1 to 1296, inclusive); read/write.

42.1.36 TextDocument.fontStyle

`textDocument.fontStyle`

Note: This functionality was added in After Effects 13.1 (CC 2014.1)

Description

String with style information, e.g., "bold", "italic"

Warning: This value only reflects the first character in the Text layer.

Type

String; read-only.

42.1.37 TextDocument.hangingRoman

`textDocument.hangingRoman`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's Roman Hanging Punctuation paragraph option. This is only meaningful to box Text layers—it allows punctuation to fit outside the box rather than flow to the next line.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.38 TextDocument.horizontalScale

`textDocument.horizontalScale`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

This Text layer's horizontal scale in pixels.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Floating-point value; read-write.

Example

```
var valOfHScale = myTextLayer.sourceText.value.horizontalScale;
```

42.1.39 TextDocument.justification

`textDocument.justification`

Description

The paragraph justification for the Text layer.

Type

A `ParagraphJustification` enumerated value; read/write. 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`
- `ParagraphJustification.MULTIPLE_JUSTIFICATIONS`

Text layers with mixed justification values will be read as `ParagraphJustification.MULTIPLE_JUSTIFICATIONS`.

Setting a TextDocument to use `ParagraphJustification.MULTIPLE_JUSTIFICATIONS` will result in `ParagraphJustification.CENTER_JUSTIFY` instead.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

42.1.40 TextDocument.kerning

`textDocument.kerning`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's kerning option.

Returns zero for `AutoKernType.METRIC_KERN` and `AutoKernType.OPTICAL_KERN`.

Setting this value will also set `AutoKernType.NO_AUTO_KERN` to true across the affected characters.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Integer value; read/write.

42.1.41 TextDocument.leading

`textDocument.leading`

Note: This functionality was added in After Effects 14.2 (CC 2017.1)

Description

The Text layer's spacing between lines.

Returns zero if *TextDocument.autoLeading* is true.

Setting this value will also set *TextDocument.autoLeading* to true across the affected characters.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

The minimum accepted value to set is 0, but this will be silently clipped to 0.01.

Type

Floating-point value; read/write.

Example

```
// This creates a Text layer and sets the leading to 100

var composition = app.project.activeItem;
var myTextLayer = comp.layers.addText("Spring\nSummer\nAutumn\nWinter");
var myTextSource = myTextLayer.sourceText;
var myTextDocument = myTextSource.value;
myTextDocument.leading = 100;
myTextSource.setValue(myTextDocument);
```

42.1.42 TextDocument.leadingType

`textDocument.leadingType`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph leading type option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

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

- `LeadingType.ROMAN_LEADING_TYPE`
- `LeadingType.JAPANESE_LEADING_TYPE`

42.1.43 TextDocument.ligature

`textDocument.ligature`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's ligature option.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.44 TextDocument.lineJoinType

`textDocument.lineJoinType`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's line join type option for Stroke.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

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

- `LineJoinType.LINE_JOIN_MITER`
- `LineJoinType.LINE_JOIN_ROUND`
- `LineJoinType.LINE_JOIN_BEVEL`

42.1.45 TextDocument.lineOrientation

`textDocument.lineOrientation`

Note: This functionality was added in After Effects 24.2.

Description

The Text layer's line orientation, in general horizontal vs vertical, which affects how all text in the layer is composed.

Type

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

- `LineOrientation.HORIZONTAL`
- `LineOrientation.VERTICAL_RIGHT_TO_LEFT`
- `LineOrientation.VERTICAL_LEFT_TO_RIGHT`

42.1.46 TextDocument.noBreak

`textDocument.noBreak`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's no break attribute.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.47 TextDocument.paragraphCount

`textDocument.paragraphCount`

Description

Returns the number of paragraphs in the text layer, always greater than or equal to 1.

Type

Number; read-only.

42.1.48 TextDocument.pointText

`textDocument.pointText`

Description

True if a Text layer is a layer of point (unbounded) text; otherwise false.

Type

Boolean; read-only.

42.1.49 TextDocument.smallCaps

`textDocument.smallCaps`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

True if a Text layer has small caps enabled; otherwise false. To set this value, use *TextDocument.fontCapsOption* added in After Effects 24.0.

Warning: This value only reflects the first character in the Text layer.

Type

Boolean; read-only.

42.1.50 TextDocument.spaceAfter

`textDocument.spaceAfter`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph space after option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Floating-point value; read/write.

42.1.51 TextDocument.spaceBefore

`textDocument.spaceBefore`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph space before option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Floating-point value; read/write.

42.1.52 TextDocument.startIndent

`textDocument.startIndent`

Note: This functionality was added in After Effects 24.0.

Description

The Text layer's paragraph start indent option.

If this attribute has a mixed value, it will be read as `undefined`.

Warning: This value reflects all paragraphs in the Text layer. If you change this value, it will set all paragraphs in the Text layer to the specified setting.

Type

Floating-point value; read/write.

42.1.53 TextDocument.strokeColor

`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.

Throws an exception on read if *applyStroke* is not true.

Setting this value will also set *applyStroke* to true across the affected characters.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Array [r, g, b] of floating-point values; read/write.

42.1.54 TextDocument.strokeOverFill

`textDocument.strokeOverFill`

Description

Indicates the rendering order for the fill and stroke of a Text layer. When true, the stroke appears over the fill.

The Text layer can override the per-character attribute setting if the Text layer is set to use All Strokes Over All Fills or All Fills Over All Strokes in the Character Panel. Thus the value returned here might be different than the actual attribute value set on the character. It is possible to set the Fill/Stroke render order via the "Fill & Stroke" property under More Options on the Text layer using *TextLayer.text("ADBE Text More Options")("ADBE Text Render Order")*.

> **Warning:** This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Boolean; read/write.

42.1.55 TextDocument.strokeWidth

`textDocument.strokeWidth`

Description

The Text layer's stroke thickness in pixels.

> **Warning:** This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.
>
> The minimum accepted value to set is 0, but this will be silently clipped to 0.01.

Type

Floating-point value (0 to 1000, inclusive); read/write.

42.1.56 TextDocument.subscript

`textDocument.subscript`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

True if a Text layer has subscript enabled; otherwise false. To set this value, use *TextDocument.fontBaselineOption* added in After Effects 24.0.

Warning: This value only reflects the first character in the Text layer.

Type

Boolean; read-only.

42.1.57 TextDocument.superscript

`textDocument.superscript`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

True if a Text layer has superscript enabled; otherwise false. To set this value, use *TextDocument.fontBaselineOption* added in After Effects 24.0.

Warning: This value only reflects the first character in the Text layer.

Type

Boolean; read-only.

42.1.58 TextDocument.text

`textDocument.text`

Description

The text value for the Text layer's Source Text property.

Type

String; read/write.

42.1.59 TextDocument.tracking

`textDocument.tracking`

Description

The Text layer's spacing between characters.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Floating-point value; read/write.

42.1.60 TextDocument.tsume

`textDocument.tsume`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

This Text layer's tsume value as a normalized percentage, from 0.0 -> 1.0.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

This attribute accepts values from 0.0 -> 100.0, however the value IS expecting a normalized value from 0.0 -> 1.0. Using a value higher than 1.0 will produce unexpected results; AE's Character Panel will clamp the value at 100%, despite the higher value set by scripting (ie `TextDocument.tsume = 100` _really_ sets a value of 10,000%)

Type

Floating-point value; read-write.

42.1.61 TextDocument.verticalScale

`textDocument.verticalScale`

Note: This functionality was added in After Effects 13.2 (CC 2014.2)

Description

This Text layer's vertical scale in pixels.

Warning: This value only reflects the first character in the Text layer. If you change this value, it will set all characters in the Text layer to the specified setting.

Type

Floating-point value; read-write.

42.2 Methods

42.2.1 TextDocument.characterRange()

```
textDocument.characterRange(characterStart, [signedCharacterEnd])
```

Note: This functionality was added in After Effects (Beta) 24.2 and is subject to change while it remains in Beta.

Description

Returns an instance of the Text layer range accessor CharacterRange.

The instance will remember the parameters passed in the constructor - they remain constant and changes to the *TextDocument* length may cause the instance to throw exceptions on access until the *TextDocument* length is changed to a length which makes the range valid again.

Use toString() to find out what the constructed parameters were.

Parameters

`characterStart`	Unsigned integer. Starts at zero, must be the less than or equal to the (text) length of the *TextDocument object*.
`signedCharacterEnd`	Optional signed integer. If not specified, will be computed at (characterStart + 1). If set to -1, then the *CharacterRange object* will dynamically calculate this on access to be equal to the (text) length of the *TextDocument object*. signedCharacterEnd must be greater than or equal to characterStart, and less than or equal to the (text) length of the *TextDocument object*.

Throws an exception if the parameters would result in an invalid range.

It is not possible to create a *CharacterRange object* which spans the final carriage return in the *TextDocument object*.

Returns

An instance of *CharacterRange object*

42.2.2 TextDocument.composedLineCharacterIndexesAt()

```
textDocument.composedLineCharacterIndexesAt(characterIndex)
```

Note: This functionality was added in After Effects (Beta) 24.3 and is subject to change while it remains in Beta.

Description

Returns the character index bounds of a *ComposedLineRange object* in the Text layer.

Parameters

characterIndex	Unsigned integer. A text index in the Text layer, which will be mapped to the composed line it intersects.

Returns

Generic object; Key `start` will be set to text index of the start of the composed line (greater than or equal to zero). Key `end` will be set to text index of the end of the composed line (greater than start, or equal to start if it is the last composed line).

Will throw an exception if the computed start and end are outside of the current *TextDocument object* Remember that the composed lines are static and subsequent changes to the *TextDocument object* instance which changes its length may render the composed line data invalid.

42.2.3 TextDocument.composedLineRange()

```
textDocument.composedLineRange(composedLineIndexStart, [signedComposedLineIndexEnd])
```

Note: This functionality was added in After Effects (Beta) 24.3 and is subject to change while it remains in Beta.

Description

Returns an instance of the Text layer range accessor *ComposedLineRange object*.

The instance will remember the parameters passed in the constructor - they remain constant and changes to the *TextDocument* contents may cause the instance to throw exceptions on access until the *TextDocument* contents are changed which makes the range valid again.

Use *ComposedLineRange.toString()* to find out what the constructed parameters were.

Parameters

composedLineIndexStart	Unsigned integer. Starts at zero, must be the less than the number of composed lines in the *TextDocument object*.
signedComposedLineIndexEnd	Optional signed integer. If not specified, will be computed at (composedLineIndexStart + 1). If set to -1, then the *ComposedLineRange object* will dynamically calculate this on access to the last composed line of the *TextDocument object*. signedComposedLineIndexEnd must be greater than composedLineIndexStart, and less than or equal to the number of composed lines in the *TextDocument object*.

Throws an exception if the parameters would result in an invalid range.

Remember that the composed lines are static and subsequent changes to the *TextDocument object* instance which changes its length may render the composed line data invalid.

Returns

An instance of *ComposedLineRange object*

42.2.4 TextDocument.paragraphCharacterIndexesAt()

```
textDocument.paragraphCharacterIndexesAt(characterIndex)
```

Note: This functionality was added in After Effects (Beta) 24.2 and is subject to change while it remains in Beta.

Description

Returns the character index bounds of a paragraph in the Text layer.

Parameters

`characterIndex`	Unsigned integer. A text index in the Text layer, which will be mapped to the paragraph it intersects.

Returns

Generic object; Key `start` will be set to text index of the start of the paragraph (greater than or equal to zero). Key end will be set to text index of the end of the paragraph (greater than start, or equal to start if it is the last paragraph).

42.2.5 TextDocument.paragraphRange()

```
textDocument.paragraphRange(paragraphIndexStart, [signedParagraphIndexEnd])
```

Note: This functionality was added in After Effects (Beta) 24.2 and is subject to change while it remains in Beta.

Description

Returns an instance of the Text layer range accessor *ParagraphRange object*.

The instance will remember the parameters passed in the constructor - they remain constant and changes to the *TextDocument* contents may cause the instance to throw exceptions on access until the *TextDocument* contents are changed which makes the range valid again.

Use *ParagraphRange.toString()* to find out what the constructed parameters were.

Parameters

`paragraphIndexStart`	Unsigned integer. Starts at zero, must be the less than the number of paragraphs in the *TextDocument object*.
`signedParagraphIndexEnd`	Optional signed integer. If not specified, will be computed at (paragraphIndexStart + 1). If set to -1, then the *ParagraphRange object* will dynamically calculate this on access to the last paragraph of the *TextDocument object*. signedParagraphIndexEnd must be greater than paragraphIndexStart, and less than or equal to the number of paragraphs in the *TextDocument object*.

Throws an exception if the parameters would result in an invalid range.

Returns

An instance of *ParagraphRange object*

42.2.6 TextDocument.resetCharStyle()

`textDocument.resetCharStyle()`

Description

Restores all characters in the Text layer to the default text character characteristics in the Character panel.

Parameters

None.

Returns

Nothing.

42.2.7 TextDocument.resetParagraphStyle()

`textDocument.resetParagraphStyle()`

Description

Restores all paragraphs in the Text layer to the default text paragraph characteristics in the Paragraph panel.

Parameters

None.

Returns

Nothing.

CHAPTER

FORTYTHREE

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.

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

43.1 Objects

- *ItemCollection object* All of the items (imported files, folders, solids, and so on) found in the Project panel.
- *LayerCollection object* All of the layers in a composition.
- *OMCollection object* All of the Output Module items in the project.
- *RQItemCollection object* All of the render-queue items in the project.

43.2 Attributes

`length`	The number of objects in the collection.

43.3 Methods

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

CHAPTER

FORTYFOUR

IMPORTOPTIONS OBJECT

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

Description

The ImportOptions object encapsulates the options used to import a file with the *Project.importFile()* methods.

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");
```

44.1 Attributes

44.1.1 ImportOptions.file

`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.

44.1.2 ImportOptions.forceAlphabetical

`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.

44.1.3 ImportOptions.importAs

`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.

Type

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

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

44.1.4 ImportOptions.rangeEnd

`importOptions.rangeEnd`

> **Warning:** This method/property is officially undocumented and was found via research. The information here may be inaccurate, and this whole method/property may disappear or stop working some point. Please contribute if you have more information on it!

Description

Sets the end clipping range of the sequence, that is going to be imported.

- Creates 'missing frames' (video-bards) if the `rangeEnd` exceeds the duration of the sequence to be imported.
- Has no effect if *sequence* is set to false.
- Throws an exception if *forceAlphabetical* is set to true.
- Throws an exception if `rangeEnd` is less then *rangeStart* and resets the range to include all the files.

Type

Integer; read/write.

44.1.5 ImportOptions.rangeStart

`importOptions.rangeStart`

> **Warning:** This method/property is officially undocumented and was found via research. The information here may be inaccurate, and this whole method/property may disappear or stop working some point. Please contribute if you have more information on it!

Description

Sets the start clipping range of the sequence, that is going to be imported.

- Has no effect if *sequence* is set to false.
- Throws an exception if *forceAlphabetical* is set to true.
- Throws an exception if *rangeEnd* value is 0.
- Throws an exception if `rangeStart` is greater then *rangeEnd* and resets the range to include all the files.

Type

Integer; read/write.

Example

```
/*
  Import 20 frames of the sequence, starting at frame 10 and ending at frame 30
 */
var mySequence = '~/Desktop/sequence/image_000.png';

var importOptions = new ImportOptions();
importOptions.file = new File(mySequence);
importOptions.sequence = true;
importOptions.forceAlphabetical = false;
importOptions.rangeStart = 10;
importOptions.rangeEnd = 30;

var item = app.project.importFile(importOptions);
```

44.1.6 ImportOptions.sequence

`importOptions.sequence`

Description

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

Type

Boolean; read/write.

44.2 Methods

44.2.1 ImportOptions.canImportAs()

`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.

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

44.2.2 ImportOptions.isFileNameNumbered()

`importOptions.isFileNameNumbered(file)`

> **Warning:** This method/property is officially undocumented and was found via research. The information here may be inaccurate, and this whole method/property may disappear or stop working some point. Please contribute if you have more information on it!

Description

Reports whether the file object is numbered, i.e. file name has a digit.

Parameters

file	Extendscript File object.

Returns

Object, containing 2 keys:

- `isNumbered`: Boolean; wether the file name contains any digit,
- `num`: Integer; a number found in file name. Returns 0 when `isNumbered` is false.

Example

```
var importOptions = new ImportOptions();
importOptions.isFileNameNumbered('image.png');     // "isNumbered": false, "num": 0
importOptions.isFileNameNumbered('003image.png');  // "isNumbered": true, "num": 3
importOptions.isFileNameNumbered('ima0102ge.png'); // "isNumbered": true, "num": 102
importOptions.isFileNameNumbered('image0120.png'); // "isNumbered": true, "num": 120
```

CHAPTER

FORTYFIVE

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

- `speed`: A floating-point value. Sets the `speed` attribute.
- `influence`: A floating-pointvalue 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]);
```

45.1 Attributes

45.1.1 KeyframeEase.influence

`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.

45.1.2 KeyframeEase.speed

`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.

CHAPTER

FORTYSIX

MARKERVALUE OBJECT

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

Description

The MarkerValue object represents a layer or composition 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 *Layer.marker* property of the layer: `layerObject.property("Marker").setValueAtTime(time, markerValueObject);`

To associate a marker with a composition, set the MarkerValue object in the *CompItem.markerProperty* property of the comp: `compObject.markerProperty.setValueAtTime(time, markerValueObject);`

For information on the usage of markers see "Using markers" in After Effects Help.

Examples

- To set a **layer** marker that says "Fade Up" at the 2 second mark:

```
var myMarker = new MarkerValue("FadeUp");
myLayer.property("Marker").setValueAtTime(2, myMarker);
// or
myLayer.marker.setValueAtTime(2, myMarker);
```

- To set a **comp** marker that says "Fade Up" at the 2 second mark:

```
var myMarker = new MarkerValue("FadeUp");
comp.markerProperty.setValueAtTime(2, myMarker);
```

- To get comment values from a particular marker:

```
var layer = app.project.item(1).layer(1);
var markerProperty = layer.marker;

var commentOfFirstMarker = markerProperty.keyValue(1).comment;

// or
var commentOfMarkerAtTime4 = markerProperty.valueAtTime(4.0, true).comment;
```

(continues on next page)

(continued from previous page)

```
// or
var markerValueAtTimeClosestToTime4 = markerProperty.keyValue(markerProperty.
↪nearestKeyIndex(4.0));
var commentOfMarkerClosestToTime4 = markerValueAtTimeClosestToTime4.comment;
```

46.1 Attributes

46.1.1 MarkerValue.chapter

`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.

46.1.2 MarkerValue.comment

`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.

46.1.3 MarkerValue.cuePointName

`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.

46.1.4 MarkerValue.duration

`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.

46.1.5 MarkerValue.eventCuePoint

`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.

46.1.6 MarkerValue.frameTarget

`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.

46.1.7 MarkerValue.url

`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.

46.1.8 MarkerValue.label

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

Description

The label color for a composition or layer marker. 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.

Available in After Effects 16.0 or later.

Type

Integer (0 to 16); read/write.

46.1.9 MarkerValue.protectedRegion

`app.project.item(index).markerProperty.keyValue(index).protectedRegion`

Description

State of the Protected Region option in the Composition Marker dialog box. When true, the composition marker behaves as a protected region. Will also return true for protected region markers on nested composition layers, but is otherwise not applicable to layer markers.

Available in After Effects 16.0 or later.

Type

Boolean; read/write.

46.2 Methods

46.2.1 MarkerValue.getParameters()

`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.

46.2.2 MarkerValue.setParameters()

`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("MyMarker");
var parms = {};
parms.timeToBlink = 1;
parms.assignMe = "A string"
mv.setParameters(parms);
myLayer.property("Marker").setValueAtTime(2, mv);
```

CHAPTER

FORTYSEVEN

PREFERENCES OBJECT

`app.preferences`

Description

The Preferences object provides an easy way to manage internal AE preferences, such as you'd find in AE's Preferences menu. These are saved in the After Effects preference files, and are persistent between application sessions.

Preferences 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 sectionname. All values are strings.

You can create new preferences with this object, as well as accessing existing preferences.

As of Version 12/CC, preferences and settings methods now take a third argument to specify the target preferences file if Section/Key is not in "Adobe After Effects $versionNumber.x Prefs.txt".

If the third argument is not passed, default value (`PREFType.PREF_Type_MACHINE_SPECIFIC`) is used and After Effects tries to save/get from the "Adobe After Effects $versionNumber.x Prefs.txt" preferences file.

The third argument is enum `PREFType` value, one of:

- `PREF_Type_MACHINE_SPECIFIC`: Adobe After Effects $versionNumber.x Prefs.txt
- `PREF_Type_MACHINE_INDEPENDENT`: Adobe After Effects $versionNumber.x Prefs-indep-general.txt
- `PREF_Type_MACHINE_INDEPENDENT_RENDER`: Adobe After Effects $versionNumber.x Prefs-indep-render.txt
- `PREF_Type_MACHINE_INDEPENDENT_OUTPUT`: Adobe After Effects $versionNumber.x Prefs-indep-output.txt
- `PREF_Type_MACHINE_INDEPENDENT_COMPOSITION`: Adobe After Effects $versionNumber.x Prefs-indep-composition.txt
- `PREF_Type_MACHINE_SPECIFIC_TEXT`: Adobe After Effects $versionNumber.x Prefs-text.txt
- `PREF_Type_MACHINE_SPECIFIC_PAINT`: Adobe After Effects $versionNumber.x Prefs-paint.txt

47.1 Methods

47.1.1 Preferences.deletePref()

`app.preferences.deletePref(sectionName, keyName[, prefType])`

Description

Deletes a preference from the preference file.

Tip: It's generally inadvised to delete internal AE preferences, however you can leverage this method to delete *Settings object* you have saved. Note that you'll need to preprend `"Settings_"` to your section name.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`prefType`	Optional, an enum indicating which preference file to use

Returns

Nothing.

Example

If you have saved a setting named with the key name "trimPrecomps" in a section called "Precomp Cropper", you can delete the setting by:

```
app.preferences.deletePref("Settings_Precomp Cropper", "trimPrecomps");
```

47.1.2 Preferences.getPrefAsBool()

`app.preferences.getPrefAsBool(sectionName, keyName[, prefType])`

Description

Retrieves a preference value from the preferences file, and parses it as a boolean.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`prefType`	Optional, an enum indicating which preference file to use

Returns

Boolean.

Example

To retrieve the value of the Flow Chart "Expand Flowchart Comps by Default" preference:

```
var expandByDefault = app.preferences.getPrefAsBool("Flowchart Settings", "Expand␣
↪Flowchart Comps by Default");
alert("The setting is: " + expandByDefault);
```

To retrieve the value of the main preference "Javascript Debugger Enabled":

```
var debuggerEnabled = app.preferences.getPrefAsBool("Main Pref Section v2", "Pref_
↪JAVASCRIPT_DEBUGGER", PREFType.PREF_Type_MACHINE_INDEPENDENT);
alert("The setting is: " + debuggerEnabled);
```

47.1.3 Preferences.getPrefAsFloat()

`app.preferences.getPrefAsFloat(sectionName, keyName[, prefType])`

Description

Retrieves a preference value from the preferences file, and parses it as a float.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`prefType`	Optional, an enum indicating which preference file to use

Returns

Float.

47.1.4 Preferences.getPrefAsLong()

`app.preferences.getPrefAsLong(sectionName, keyName[, prefType])`

Description

Retrieves a preference value from the preferences file, and parses it as a long (number).

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`prefType`	Optional, an enum indicating which preference file to use

Returns

Long.

47.1.5 Preferences.getPrefAsString()

`app.preferences.getPrefAsString(sectionName, keyName[, prefType])`

Description

Retrieves a preference value from the preferences file, and parses it as a string.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`prefType`	Optional, an enum indicating which preference file to use

Returns

String.

47.1.6 Preferences.havePref()

`app.preferences.havePref(sectionName, keyName[, prefType])`

Description

Returns true if the specified preference item exists and has a value.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`prefType`	Optional, an enum indicating which preference file to use

Returns

Boolean.

47.1.7 Preferences.reload()

`app.preferences.reload()`

Description

Reloads the preferences file manually. Otherwise, changes to preferences will only be accessible by scripting after an application restart.

Parameters

None.

Returns

Nothing.

47.1.8 Preferences.savePrefAsBool()

`app.preferences.savePrefAsBool(sectionName, keyName, value[, prefType])`

Description

Saves a preference item as a boolean.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`value`	A boolean containing the new value
`prefType`	Optional, an enum indicating which preference file to use

Returns

Nothing.

47.1.9 Preferences.savePrefAsFloat()

`app.preferences.savePrefAsFloat(sectionName, keyName, value[, prefType])`

Description

Saves a preference item as a float.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`value`	A float containing the new value
`prefType`	Optional, an enum indicating which preference file to use

Returns

Nothing.

47.1.10 Preferences.savePrefAsLong()

`app.preferences.savePrefAsLong(sectionName, keyName, value[, prefType])`

Description

Saves a preference item as a long.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`value`	A long containing the new value
`prefType`	Optional, an enum indicating which preference file to use

Returns

Nothing.

47.1.11 Preferences.savePrefAsString()

`app.preferences.savePrefAsString(sectionName, keyName, value[, prefType])`

Description

Saves a preference item as a string.

Parameters

`sectionName`	A string containing the name of a preferences section
`keyName`	A string containing the key name of the preference
`value`	A string containing the new value
`prefType`	Optional, an enum indicating which preference file to use

Returns

Nothing.

47.1.12 Preferences.saveToDisk()

`app.preferences.saveToDisk()`

Description

Saves the preferences to disk manually. Otherwise, changes to preferences will only be accessible by scripting after an application restart.

Parameters

None.

Returns

Nothing.

CHAPTER

FORTYEIGHT

SETTINGS OBJECT

`app.settings`

Description

The Settings object provides an easy way to manage settings for third-party scripts. The settings are saved in the main 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 settings file, section names are enclosed in brackets and quotation marks, and key names are listing in quotation marks below the sectionname. All values are strings.

You can create new settings with this object, as well as accessing existing settings.

As of Version 12/CC, preferences and settings methods now take a third argument to specify the target preferences file if Section/Key is not in the main preferences file. See *Preferences object* for more info.

Note:

- These values aren't shared between versions of AE; each new install brings new settings files, and so these prefs won't carry over.
- Internally, all saved settings have their section name preprended with `"Settings_"`
- If you're looking to get or set internal AE preferences, see *Preferences object*

Tip:

- It's best practice to use one `sectionName` per script; this keeps your settings organized and easy to find & work with.
- There isn't really any benefit in saving your settings to a specific preferences file; omitting the third argument and using the default is likely all you'll need.

48.1 Methods

48.1.1 Settings.getSetting()

`app.settings.getSetting(sectionName, keyName[, prefType])`

Description

Retrieves a script settings item value from the preferences file.

> **Warning:** If the value is greater than 1999 bytes, `getSetting` that item will throw an error (seen in AE 15.0.1)

Parameters

`sectionName`	A string containing the name of a settings section.
`keyName`	A string containing the key name of the setting item.
`prefType`	Optional, an enum indicating which preference file to use.

Returns

String.

Example

If you have saved a setting named with the key name "trimPrecomps" in a section called "Precomp Cropper", you can retrieve the value by:

```
var trimPrecompsSetting = app.settings.getSetting("Precomp Cropper", "trimPrecomps");
alert("The setting is: " + trimPrecompsSetting);
```

48.1.2 Settings.haveSetting()

`app.settings.haveSetting(sectionName, keyName[, prefType])`

Description

Returns true if the specified script settings 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.
`prefType`	Optional, an enum indicating which preference file to use.

Returns

Boolean.

48.1.3 Settings.saveSetting()

`app.settings.saveSetting(sectionName, keyName, value[, prefType])`

Description

Saves a value for a script settings item.

> **Warning:** If the value is greater than 1999 bytes, `saveSetting` that item will throw an error (seen in AE 15.0.1)

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.
`prefType`	Optional, an enum indicating which preference file to use.

Returns

Nothing.

Example

If you want to save a setting called "trimPrecomps" for a script named "Precomp Cropper", you could save that setting via

```
var trimPrecompsSetting = true;
app.settings.saveSetting("Precomp Cropper", "trimPrecomps", trimPrecompsSetting);
```

Note that the setting will be saved as a string. You'll want to parse it into a bool later, if needed.

CHAPTER

FORTYNINE

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*) 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.

Examples

- 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;
```

- 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;
```

- 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;
```

- 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 counter clockwise
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 (onright-
↪sidesegment) has a larger radius
```

49.1 Attributes

49.1.1 Shape.closed

`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.

49.1.2 Shape.featherInterps

`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.

49.1.3 Shape.featherRadii

`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.

49.1.4 Shape.featherRelCornerAngles

`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.

49.1.5 Shape.featherRelSegLocs

`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.

49.1.6 Shape.featherSegLocs

`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);
```

49.1.7 Shape.featherTensions

`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.

49.1.8 Shape.featherTypes

`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.

49.1.9 Shape.inTangents

`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.

49.1.10 Shape.outTangents

`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.

49.1.11 Shape.vertices

`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.

Example

```
myShape.vertices = [[0,0], [0,1], [1,1], [1,0]];
```

Type

Array of floating-point pair arrays; read/write.

CHAPTER

FIFTY

VIEW OBJECT

`app.activeViewer.views[0]`

Description

The View object represents a specific view.

50.1 Attributes

50.1.1 View.active

`app.activeViewer.views[0].active`

Description

When true, indicates if the viewer panel is focused, and thereby frontmost.

Type

Boolean; read-only.

50.1.2 View.options

`app.activeViewer.views[0].options`

Description

Options object for this View

Type

ViewOptions object

50.2 Methods

50.2.1 View.setActive()

```
app.activeViewer.views[0].setActive()
```

Description

Moves this view panel to the front and places focus on it, making it active. Calling this method will set the *view's active attribute* to true.

Parameters

None.

Returns

Boolean, indicating if the view panel was made active.

CHAPTER

FIFTYONE

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.");
}
```

51.1 Attributes

51.1.1 Viewer.active

`viewer.active`

Description

When true, indicates if the viewer panel is focused, and thereby frontmost.

Type

Boolean; read-only.

51.1.2 Viewer.activeViewIndex

`viewer.activeViewIndex`

Description

The index of the current active *View object*, in the *Viewer.views* array.

Type

Number; read/write.

51.1.3 Viewer.maximized

`viewer.maximized`

Description

When true, indicates if the viewer panel is at its maximized size.

Type

Boolean; read/write.

51.1.4 Viewer.views

`viewer.views`

Description

All of the Views associated with this viewer.

Type

Array of *View object* objects; read-only.

51.1.5 Viewer.type

`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`

51.2 Methods

51.2.1 Viewer.setActive()

`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.

CHAPTER

FIFTYTWO

VIEWOPTIONS OBJECT

`app.activeViewer.views[0].options`

Description

The ViewOptions object represents the options for a given *View object*

Example

This enables checkerboards and locks guides for a given view

```
var activeViewer = app.activeViewer;
var viewOptions = activeViewer.views[0].options;
viewOptions.checkerboards = true;
viewOptions.guidesLocked = true;
```

52.1 Attributes

52.1.1 ViewOptions.channels

`app.activeViewer.views[0].options.channels`

Description

The state of the Channels menu.

Type

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

- `CHANNEL_ALPHA`
- `CHANNEL_ALPHA_BOUNDARY`
- `CHANNEL_ALPHA_OVERLAY`
- `CHANNEL_BLUE`
- `CHANNEL_BLUE_COLORIZE`
- `CHANNEL_GREEN`
- `CHANNEL_GREEN_COLORIZE`
- `CHANNEL_RED`
- `CHANNEL_RED_COLORIZE`

- CHANNEL_RGB
- CHANNEL_RGB_STRAIGHT

52.1.2 ViewOptions.checkerboards

app.activeViewer.views[0].options.checkerboards

Description

When true, checkerboards (transparency grid) is enabled in the current view.

Type

Boolean; read/write.

52.1.3 ViewOptions.exposure

app.activeViewer.views[0].options.exposure

Description

The exposure value for the current view.

Type

Floating-point value in the range [-40..40]

52.1.4 ViewOptions.fastPreview

app.activeViewer.views[0].options.fastPreview

Note: This functionality was added in After Effects 12.0 (CC)

Description

The state of the Fast Previews menuThis is a read/write attribute using an enumerated value:

Warning: If you try to get or set the attribute's value in the Layer or Footage panel, you'll get an error message.

Note: The Draft preview mode is only available in ray-traced 3D compositions. If you try to use it in a Classic 3D composition, you'll get an error: "Cannot set Draft fast preview mode in a Classic 3D composition."

Type

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

- FastPreviewType.FP_OFF: Off (Final Quality)

- `FastPreviewType.FP_ADAPTIVE_RESOLUTION`: Adaptive Resolution
- `FastPreviewType.FP_DRAFT`: Draft
- `FastPreviewType.FP_FAST_DRAFT`: Fast Draft
- `FastPreviewType.FP_WIREFRAME`: Wireframe

Example

```
app.activeViewer.views[0].options.fastPreview === FastPreviewType.FP_ADAPTIVE_RESOLUTION;
app.activeViewer.views[0].options.fastPreview === FastPreviewType.FP_DRAFT;
app.activeViewer.views[0].options.fastPreview === FastPreviewType.FP_FAST_DRAFT;
app.activeViewer.views[0].options.fastPreview === FastPreviewType.FP_OFF;
app.activeViewer.views[0].options.fastPreview === FastPreviewType.FP_WIREFRAME;
```

52.1.5 ViewOptions.guidesLocked

`app.activeViewer.views[0].options.guidesLocked`

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

When true, indicates guides are locked in the view.

Type

Boolean; read/write.

Example

```
app.activeViewer.views[0].options.guidesLocked;
```

52.1.6 ViewOptions.guidesSnap

`app.activeViewer.views[0].options.guidesSnap`

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

When true, indicates layers snap to guides when dragged in the view.

Type

Boolean; read/write.

Example

```
app.activeViewer.views[0].options.guidesSnap;
```

52.1.7 ViewOptions.guidesVisibility

`app.activeViewer.views[0].options.guidesVisibility`

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

When true, indicates guides are visible in the view.

Type

Boolean; read/write.

Example

```
app.activeViewer.views[0].options.guidesVisibility;
```

52.1.8 ViewOptions.rulers

`app.activeViewer.views[0].options.rulers`

Note: This functionality was added in After Effects 16.1 (CC 2019)

Description

When true, indicates rulers are shown in the view.

Type

Boolean; read/write.

Example

```
app.activeViewer.views[0].options.rulers;
```

52.1.9 ViewOptions.zoom

`app.activeViewer.views[0].options.zoom`

Description

Sets the current zoom value for the view, as a normalized percentage between 1% (*0.01*) and 1600% (*16*).

Type

Floating-point value in the range `[0.01..16]`

CHAPTER

FIFTYTHREE

AVLAYER MATCH NAMES

Category	Match Name	Display Name (EN)
Layer	ADBE AV Layer	
Top-Level	ADBE Marker	Marker
	ADBE Time Remapping	Time Remap
	ADBE MTrackers	Motion Trackers
	ADBE Mask Parade	Masks
	ADBE Effect Parade	Effects
	ADBE Layer Overrides	Essential Properties
Transform	ADBE Transform Group	Transform
	ADBE Anchor Point	Anchor Point
	ADBE Position	Position
	ADBE Position_0	X Position
	ADBE Position_1	Y Position
	ADBE Position_2	Z Position
	ADBE Scale	Scale
	ADBE Orientation	Orientation
	ADBE Rotate X	X Rotation
	ADBE Rotate Y	Y Rotation
	ADBE Rotate Z	Z Rotation
	ADBE Opacity	Opacity
Audio	ADBE Audio Group	Audio
	ADBE Audio Levels	Audio Levels
Essential Properties	ADBE Layer Source Alternate	

CHAPTER

FIFTYFOUR

3D LAYER MATCH NAMES

Category	Match Name	Display Name (EN)
Plane	ADBE Plane Options Group	Geometry Options
	ADBE Plane Curvature	Curvature
	ADBE Plane Subdivision	Segments
	ADBE Extrsn Options Group	Geometry Options
	ADBE Bevel Depth	Bevel Depth
	ADBE Hole Bevel Depth	Hole Bevel Depth
	ADBE Extrsn Depth	Extrusion Depth
Materials	ADBE Material Options Group	Material Options
	ADBE Light Transmission	Light Transmission
	ADBE Ambient Coefficient	Ambient
	ADBE Diffuse Coefficient	Diffuse
	ADBE Specular Coefficient	Specular Intensity
	ADBE Shininess Coefficient	Specular Shininess
	ADBE Metal Coefficient	Metal
	ADBE Reflection Coefficient	Reflection Intensity
	ADBE Glossiness Coefficient	Reflection Sharpness
	ADBE Fresnel Coefficient	Reflection Rolloff
	ADBE Transparency Coefficient	Transparency
	ADBE Transp Rolloff	Transparency Rolloff
	ADBE Index of Refraction	Index of Refraction

CHAPTER

FIFTYFIVE

CAMERA LAYER MATCH NAMES

Category	Match Name	Display Name (EN)
Layer	ADBE Camera Layer	
Camera	ADBE Camera Options Group	Camera Options
	ADBE Camera Zoom	Zoom
	ADBE Camera Depth of Field	Depth of Field
	ADBE Camera Focus Distance	Focus Distance
	ADBE Camera Aperture	Aperture
	ADBE Camera Blur Level	Blur Level
Iris	ADBE Iris Shape	Iris Shape
	ADBE Iris Rotation	Iris Rotation
	ADBE Iris Roundness	Iris Roundness
	ADBE Iris Aspect Ratio	Iris Aspect Ratio
	ADBE Iris Diffraction Fringe	Iris Diffraction Fringe
	ADBE Iris Highlight Gain	Highlight Gain
	ADBE Iris Highlight Threshold	Highlight Threshold
	ADBE Iris Hightlight Saturation	Highlight Saturation

CHAPTER

FIFTYSIX

LIGHT LAYER MATCH NAMES

Category	Match Name	Display Name (EN)
Layer	`ADBE Light Layer`	
Light	`ADBE Light Options Group`	Light Options
	`ADBE Light Intensity`	Intensity
	`ADBE Light Color`	Color
	`ADBE Light Cone Angle`	Cone Angle
	`ADBE Light Cone Feather 2`	Cone Feather
Falloff	`ADBE Light Falloff Type`	Falloff
	`ADBE Light Falloff Start`	Radius
	`ADBE Light Falloff Distance`	Falloff Distance
Shadow	`ADBE Light Shadow Darkness`	Shadow Darkness
	`ADBE Light Shadow Diffusion`	Shadow Diffusion

CHAPTER

FIFTYSEVEN

TEXT LAYER MATCH NAMES

Category	Match Name	Display Name (EN)
Layer	ADBE Text Layer	
Text	ADBE Text Properties	Text
	ADBE Text Document	Source Text
	ADBE Text Path Options	Path Options
	ADBE Text Reverse Path	Reverse Path
	ADBE Text Perpendicular To Path	Perpendicular To Path
	ADBE Text Force Align Path	Force Alignment
	ADBE Text First Margin	First Margin
	ADBE Text Last Margin	Last Margin
	ADBE Text More Options	More Options
	ADBE Text Anchor Point Align	Grouping Alignment
	ADBE Text Animators	Animators
Animators	ADBE Text Animator	Animator
	ADBE Text Selectors	Selectors
	ADBE Text Selector	Range Selector
	ADBE Text Percent Start	Start
	ADBE Text Percent End	End
	ADBE Text Percent Offset	Offset
	ADBE Text Index Start	Start
	ADBE Text Index End	End
	ADBE Text Index Offset	Offset
	ADBE Text Range Advanced	Advanced
	ADBE Text Selector Mode	Mode
	ADBE Text Selector Max Amount	Amount
	ADBE Text Selector Smoothness	Smoothness
	ADBE Text Levels Max Ease	Ease High
	ADBE Text Levels Min Ease	Ease Low
	ADBE Text Random Seed	Random Seed
	ADBE Text Animator Properties	Properties
	ADBE Text Anchor Point 3D	Anchor Point
	ADBE Text Position 3D	Position
	ADBE Text Scale 3D	Scale
	ADBE Text Skew	Skew
	ADBE Text Skew Axis	Skew Axis
	ADBE Text Rotation X	X Rotation
	ADBE Text Rotation Y	Y Rotation

continues on next page

Table 1 – continued from previous page

	ADBE Text Rotation	Z Rotation
	ADBE Text Opacity	Opacity
	ADBE Text Fill Opacity	Fill Opacity
	ADBE Text Stroke Opacity	Stroke Opacity
	ADBE Text Fill Color	Fill Color
	ADBE Text Stroke Color	Stroke Color
	ADBE Text Fill Hue	Fill Hue
	ADBE Text Stroke Hue	Stroke Hue
	ADBE Text Fill Saturation	Fill Saturation
	ADBE Text Stroke Saturation	Stroke Saturation
	ADBE Text Fill Brightness	Fill Brightness
	ADBE Text Stroke Brightness	Stroke Brightness
	ADBE Text Stroke Width	Stroke Width
	ADBE Text Line Anchor	Line Anchor
	ADBE Text Track Type	Tracking Type
	ADBE Text Tracking Amount	Tracking Amount
	ADBE Text Character Replace	Character Value
	ADBE Text Character Offset	Character Offset
	ADBE Text Line Spacing	Line Spacing
	ADBE Text Blur	Blur
3d Text	ADBE 3DText Front RGB	Front Color
	ADBE 3DText Front Hue	Front Hue
	ADBE 3DText Front Sat	Front Saturation
	ADBE 3DText Front Bright	Front Brightness
	ADBE 3DText Front Opacity	Front Opacity
	ADBE 3DText Front Ambient	Front Ambient
	ADBE 3DText Front Diffuse	Front Diffuse
	ADBE 3DText Front Specular	Front Specular Intensity
	ADBE 3DText Front Shininess	Front Specular Shininess
	ADBE 3DText Front Metal	Front Metal
	ADBE 3DText Front Reflection	Front Reflection Intensity
	ADBE 3DText Front Gloss	Front Reflection Sharpness
	ADBE 3DText Front Fresnel	Front Reflection Rolloff
	ADBE 3DText Front Xparency	Front Transparency
	ADBE 3DText Front XparRoll	Front Transparency Rolloff
	ADBE 3DText Front IOR	Front Index of Refraction
	ADBE 3DText Bevel RGB	Bevel Color
	ADBE 3DText Bevel Hue	Bevel Hue
	ADBE 3DText Bevel Sat	Bevel Saturation
	ADBE 3DText Bevel Bright	Bevel Brightness
	ADBE 3DText Bevel Opacity	Bevel Opacity
	ADBE 3DText Bevel Ambient	Bevel Ambient
	ADBE 3DText Bevel Diffuse	Bevel Diffuse
	ADBE 3DText Bevel Specular	Bevel Specular Intensity
	ADBE 3DText Bevel Shininess	Bevel Specular Shininess
	ADBE 3DText Bevel Metal	Bevel Metal
	ADBE 3DText Bevel Reflection	Bevel Reflection Intensity
	ADBE 3DText Bevel Gloss	Bevel Reflection Sharpness
	ADBE 3DText Bevel Fresnel	Bevel Reflection Rolloff
	ADBE 3DText Bevel Xparency	Bevel Transparency
	ADBE 3DText Bevel XparRoll	Bevel Transparency Rolloff

continues on next page

Table 1 – continued from previous page

	ADBE 3DText Bevel IOR	Bevel Index of Refraction
	ADBE 3DText Side RGB	Side Color
	ADBE 3DText Side Hue	Side Hue
	ADBE 3DText Side Sat	Side Saturation
	ADBE 3DText Side Bright	Side Brightness
	ADBE 3DText Side Opacity	Side Opacity
	ADBE 3DText Side Ambient	Side Ambient
	ADBE 3DText Side Diffuse	Side Diffuse
	ADBE 3DText Side Specular	Side Specular Intensity
	ADBE 3DText Side Shininess	Side Specular Shininess
	ADBE 3DText Side Metal	Side Metal
	ADBE 3DText Side Reflection	Side Reflection Intensity
	ADBE 3DText Side Gloss	Side Reflection Sharpness
	ADBE 3DText Side Fresnel	Side Reflection Rolloff
	ADBE 3DText Side Xparency	Side Transparency
	ADBE 3DText Side XparRoll	Side Transparency Rolloff
	ADBE 3DText Side IOR	Side Index of Refraction
	ADBE 3DText Back RGB	Back Color
	ADBE 3DText Back Hue	Back Hue
	ADBE 3DText Back Sat	Back Saturation
	ADBE 3DText Back Bright	Back Brightness
	ADBE 3DText Back Opacity	Back Opacity
	ADBE 3DText Back Ambient	Back Ambient
	ADBE 3DText Back Diffuse	Back Diffuse
	ADBE 3DText Back Specular	Back Specular Intensity
	ADBE 3DText Back Shininess	Back Specular Shininess
	ADBE 3DText Back Metal	Back Metal
	ADBE 3DText Back Reflection	Back Reflection Intensity
	ADBE 3DText Back Gloss	Back Reflection Sharpness
	ADBE 3DText Back Fresnel	Back Reflection Rolloff
	ADBE 3DText Back Xparency	Back Transparency
	ADBE 3DText Back XparRoll	Back Transparency Rolloff
	ADBE 3DText Back IOR	Back Index of Refraction
	ADBE 3DText Bevel Depth	Bevel Depth
	ADBE 3DText Extrude Depth	Extrusion Depth

CHAPTER

FIFTYEIGHT

SHAPE LAYER MATCH NAMES

Category	Match Name	Display Name (EN)
Layer	ADBE Vector Layer	Shape Layer
Contents	ADBE Root Vectors Group	Contents
Group	ADBE Vector Group	Group
	ADBE Vector Blend Mode	Blend Mode
	ADBE Vectors Group	Contents
	ADBE Vector Transform Group	Transform
	ADBE Vector Materials Group	Material Options
Rectangle	ADBE Vector Shape - Rect	Rectangle
	ADBE Vector Shape Direction	Shape Direction
	ADBE Vector Rect Size	Size
	ADBE Vector Rect Position	Position
	ADBE Vector Rect Roundness	Roundness
Ellipse	ADBE Vector Shape - Ellipse	Ellipse
	ADBE Vector Shape Direction	Shape Direction
	ADBE Vector Ellipse Size	Size
	ADBE Vector Ellipse Position	Position
Polystar	ADBE Vector Shape - Star	Polystar
	ADBE Vector Shape Direction	Shape Direction
	ADBE Vector Star Type	Type
	ADBE Vector Star Points	Points
	ADBE Vector Star Position	Position
	ADBE Vector Star Rotation	Rotation
	ADBE Vector Star Inner Radius	Inner Radius
	ADBE Vector Star Outer Radius	Outer Radius
	ADBE Vector Star Inner Roundess	Inner Roundness
	ADBE Vector Star Outer Roundess	Outer Roundness
Path	ADBE Vector Shape - Group	Path
	ADBE Vector Shape Direction	Shape Direction
	ADBE Vector Shape	Path
Fill	ADBE Vector Graphic - Fill	Fill
	ADBE Vector Blend Mode	Blend Mode

continues on next page

Table 1 – continued from previous page

Category	Match Name	Display Name (EN)
	ADBE Vector Composite Order	Composite
	ADBE Vector Fill Rule	Fill Rule
	ADBE Vector Fill Color	Color
	ADBE Vector Fill Opacity	Opacity
Stroke	ADBE Vector Graphic - Stroke	Stroke
	ADBE Vector Blend Mode	Blend Mode
	ADBE Vector Composite Order	Composite
	ADBE Vector Stroke Color	Color
	ADBE Vector Stroke Opacity	Opacity
	ADBE Vector Stroke Width	Stroke Width
	ADBE Vector Stroke Line Cap	Line Cap
	ADBE Vector Stroke Line Join	Line Join
	ADBE Vector Stroke Miter Limit	Miter Limit
Stroke Dashes	ADBE Vector Stroke Dashes	Dashes
	ADBE Vector Stroke Dash 1	Dash
	ADBE Vector Stroke Gap 1	Gap
	ADBE Vector Stroke Dash 2	Dash 2
	ADBE Vector Stroke Gap 2	Gap 2
	ADBE Vector Stroke Dash 3	Dash 3
	ADBE Vector Stroke Gap 3	Gap 3
	ADBE Vector Stroke Offset	Offset
Stroke Taper	ADBE Vector Stroke Taper	Taper
	ADBE Vector Taper Start Width	Start Width
	ADBE Vector Taper Length Units	Length Units
	ADBE Vector Taper End Width	End Width
	ADBE Vector Taper End Ease	End Ease
	ADBE Vector Taper End Length	End Length
	ADBE Vector Taper Start Length	Start Length
	ADBE Vector Taper Start Ease	Start Ease
Stroke Wave	ADBE Vector Stroke Wave	Wave
	ADBE Vector Taper Wave Amount	Amount
	ADBE Vector Taper Wave Units	Units
	ADBE Vector Taper Wave Phase	Phase
	ADBE Vector Taper Wavelength	Wavelength
Gradient Fill	ADBE Vector Graphic - G-Fill	Gradient Fill
	ADBE Vector Blend Mode	Blend Mode
	ADBE Vector Composite Order	Composite
	ADBE Vector Fill Rule	Fill Rule
	ADBE Vector Grad Type	Type
	ADBE Vector Grad Start Pt	Start Point
	ADBE Vector Grad End Pt	End Point
	ADBE Vector Grad HiLite Length	Highlight Length
	ADBE Vector Grad HiLite Angle	Highlight Angle
	ADBE Vector Grad Colors	Colors
	ADBE Vector Fill Opacity	Opacity

continues on next page

Table 1 – continued from previous page

Category	Match Name	Display Name (EN)
Gradient Stroke	ADBE Vector Graphic - G-Stroke	Gradient Stroke
	ADBE Vector Blend Mode	Blend Mode
	ADBE Vector Composite Order	Composite
	ADBE Vector Grad Type	Type
	ADBE Vector Grad Start Pt	Start Point
	ADBE Vector Grad End Pt	End Point
	ADBE Vector Grad HiLite Length	Highlight Length
	ADBE Vector Grad HiLite Angle	Highlight Angle
	ADBE Vector Grad Colors	Colors
	ADBE Vector Stroke Opacity	Opacity
	ADBE Vector Stroke Width	Stroke Width
	ADBE Vector Stroke Line Cap	Line Cap
	ADBE Vector Stroke Line Join	Line Join
	ADBE Vector Stroke Miter Limit	Miter Limit
	ADBE Vector Stroke Dashes	Dashes
Merge Paths	ADBE Vector Filter - Merge	Merge Paths
	ADBE Vector Merge Type	Mode
Offset Paths	ADBE Vector Filter - Offset	Offset Paths
	ADBE Vector Offset Amount	Amount
	ADBE Vector Offset Line Join	Line Join
	ADBE Vector Offset Miter Limit	Miter Limit
	ADBE Vector Offset Copies	Copies
	ADBE Vector Offset Copy Offset	Copy Offset
Pucker & Bloat	ADBE Vector Filter - PB	Pucker & Bloat
	ADBE Vector PuckerBloat Amount	Amount
Repeater	ADBE Vector Filter - Repeater	Repeater
	ADBE Vector Repeater Copies	Copies
	ADBE Vector Repeater Offset	Offset
	ADBE Vector Repeater Order	Composite
	ADBE Vector Repeater Transform	Transform
Round Corners	ADBE Vector Filter - RC	Round Corners
	ADBE Vector RoundCorner Radius	Radius
Trim Paths	ADBE Vector Filter - Trim	Trim Paths
	ADBE Vector Trim Start	Start
	ADBE Vector Trim End	End
	ADBE Vector Trim Offset	Offset
	ADBE Vector Trim Type	Trim Multiple Shapes
Twist	ADBE Vector Filter - Twist	Twist
	ADBE Vector Twist Angle	Angle
	ADBE Vector Twist Center	Center
Wiggle Paths	ADBE Vector Filter - Roughen	Wiggle Paths
	ADBE Vector Roughen Size	Size
	ADBE Vector Roughen Detail	Detail

continues on next page

Table 1 – continued from previous page

Category	Match Name	Display Name (EN)
	ADBE Vector Roughen Points	Points
	ADBE Vector Temporal Freq	Wiggles/Second
	ADBE Vector Correlation	Correlation
	ADBE Vector Temporal Phase	Temporal Phase
	ADBE Vector Spatial Phase	Spatial Phase
	ADBE Vector Random Seed	Random Seed
Wiggle Transform	ADBE Vector Filter - Wiggler	Wiggle Transform
	ADBE Vector Xform Temporal Freq	Wiggles/Second
	ADBE Vector Correlation	Correlation
	ADBE Vector Temporal Phase	Temporal Phase
	ADBE Vector Spatial Phase	Spatial Phase
	ADBE Vector Random Seed	Random Seed
	ADBE Vector Wiggler Transform	Transform
Zig Zag	ADBE Vector Filter - Zigzag	Zig Zag
	ADBE Vector Zigzag Size	Size
	ADBE Vector Zigzag Detail	Ridges per segment
	ADBE Vector Zigzag Points	Points

CHAPTER

FIFTYNINE

LAYER STYLES MATCH NAMES

Category	Match Name	Display Name (EN)
Styles	ADBE Blend Options Group	Blending Options
	ADBE Global Angle2	Global Light Angle
	ADBE Global Altitude2	Global Light Altitude
	ADBE Adv Blend Group	Advanced Blending
	ADBE Layer Fill Opacity2	Fill Opacity
	ADBE R Channel Blend	Red
	ADBE G Channel Blend	Green
	ADBE B Channel Blend	Blue
	ADBE Blend Interior	Blend Interior Styles as Group
	ADBE Blend Ranges	Use Blend Ranges from Source
Drop Shadow	dropShadow/enabled	Drop Shadow
	dropShadow/mode2	Blend Mode
	dropShadow/color	Color
	dropShadow/opacity	Opacity
	dropShadow/useGlobalAngle	Use Global Light
	dropShadow/localLightingAngle	Angle
	dropShadow/distance	Distance
	dropShadow/chokeMatte	Spread
	dropShadow/blur	Size
	dropShadow/noise	Noise
	dropShadow/layerConceals	Layer Knocks Out Drop Shadow
Inner Shadow	innerShadow/enabled	Inner Shadow
	innerShadow/mode2	Blend Mode
	innerShadow/color	Color
	innerShadow/opacity	Opacity
	innerShadow/useGlobalAngle	Use Global Light
	innerShadow/localLightingAngle	Angle
	innerShadow/distance	Distance
	innerShadow/chokeMatte	Choke
	innerShadow/blur	Size
	innerShadow/noise	Noise
Outer Glow	outerGlow/enabled	Outer Glow
	outerGlow/mode2	Blend Mode
	outerGlow/opacity	Opacity
	outerGlow/noise	Noise

continues on next page

Table 1 – continued from previous page

	outerGlow/AEColorChoice	Color Type
	outerGlow/color	Color
	outerGlow/gradient	Colors
	outerGlow/gradientSmoothness	Gradient Smoothness
	outerGlow/glowTechnique	Technique
	outerGlow/chokeMatte	Spread
	outerGlow/blur	Size
	outerGlow/inputRange	Range
	outerGlow/shadingNoise	Jitter
Inner Glow	innerGlow/enabled	Inner Glow
	innerGlow/mode2	Blend Mode
	innerGlow/opacity	Opacity
	innerGlow/noise	Noise
	innerGlow/AEColorChoice	Color Type
	innerGlow/color	Color
	innerGlow/gradient	Colors
	innerGlow/gradientSmoothness	Gradient Smoothness
	innerGlow/glowTechnique	Technique
	innerGlow/innerGlowSource	Source
	innerGlow/chokeMatte	Choke
	innerGlow/blur	Size
	innerGlow/inputRange	Range
	innerGlow/shadingNoise	Jitter
Bevel/Emboss	bevelEmboss/enabled	Bevel and Emboss
	bevelEmboss/bevelStyle	Style
	bevelEmboss/bevelTechnique	Technique
	bevelEmboss/strengthRatio	Depth
	bevelEmboss/bevelDirection	Direction
	bevelEmboss/blur	Size
	bevelEmboss/softness	Soften
	bevelEmboss/useGlobalAngle	Use Global Light
	bevelEmboss/localLightingAngle	Angle
	bevelEmboss/localLightingAltitude	Altitude
	bevelEmboss/highlightMode	Highlight Mode
	bevelEmboss/highlightColor	Highlight Color
	bevelEmboss/highlightOpacity	Highlight Opacity
	bevelEmboss/shadowMode	Shadow Mode
	bevelEmboss/shadowColor	Shadow Color
	bevelEmboss/shadowOpacity	Shadow Opacity
Satin	chromeFX/enabled	Satin
	chromeFX/mode2	Blend Mode
	chromeFX/color	Color
	chromeFX/opacity	Opacity
	chromeFX/localLightingAngle	Angle
	chromeFX/distance	Distance
	chromeFX/blur	Size
	chromeFX/invert	Invert
Solid Fill	solidFill/enabled	Color Overlay

continues on next page

Table 1 – continued from previous page

	solidFill/mode2	Blend Mode
	solidFill/color	Color
	solidFill/opacity	Opacity
Grad Fill	gradientFill/enabled	Gradient Overlay
	gradientFill/mode2	Blend Mode
	gradientFill/opacity	Opacity
	gradientFill/gradient	Colors
	gradientFill/gradientSmoothness	Gradient Smoothness
	gradientFill/angle	Angle
	gradientFill/type	Style
	gradientFill/reverse	Reverse
	gradientFill/align	Align with Layer
	gradientFill/scale	Scale
	gradientFill/offset	Offset
Pattern	patternFill/enabled	Pattern Overlay
	patternFill/mode2	Blend Mode
	patternFill/opacity	Opacity
	patternFill/align	Link with Layer
	patternFill/scale	Scale
	patternFill/phase	Offset
Stroke	frameFX/enabled	Stroke
	frameFX/mode2	Blend Mode
	frameFX/color	Color
	frameFX/size	Size
	frameFX/opacity	Opacity
	frameFX/style	Position

CHAPTER

SIXTY

FIRST-PARTY EFFECT MATCH NAMES

This list also details effect Bits Per Channel (BPC) and the AE version GPU-acceleration was introduced, if applicable.

Category	Match Name	Display Name (EN)	BPC	GPU
3D Channel	ADBE AUX CHANNEL EXTRACT	3D Channel Extract	8	
	ADBE DEPTH MATTE	Depth Matte	32	
	ADBE DEPTH FIELD	Depth of Field	32	
	EXtractoR	EXtractoR	32	
	ADBE FOG_3D	Fog 3D	32	
	ADBE ID MATTE	ID Matte	32	
	IDentifier	IDentifier	32	
Audio	ADBE Aud Reverse	Backwards		
	ADBE Aud BT	Bass & Treble		
	ADBE Aud Delay	Delay		
	ADBE Aud_Flange	Flange & Chorus		
	ADBE Aud HiLo	High-Low Pass		
	ADBE Aud Modulator	Modulator		
	ADBE Param EQ	Parametric EQ		
	ADBE Aud Reverb	Reverb		
	ADBE Aud Stereo Mixer	Stereo Mixer		
	ADBE Aud Tone	Tone		
Blur & Sharpen	ADBE Bilateral	Bilateral Blur	32	
	ADBE Camera Lens Blur	Camera Lens Blur	32	
	ADBE CameraShakeDeblur	Camera-Shake Deblur	32	
	CS CrossBlur	CC Cross Blur	32	
	CC Radial Blur	CC Radial Blur	32	
	CC Radial Fast Blur	CC Radial Fast Blur	16	
	CC Vector Blur	CC Vector Blur	16	
	ADBE Channel Blur	Channel Blur	32	
	ADBE Compound Blur	Compound Blur	32	
	ADBE Motion Blur	Directional Blur	32	15.0
	ADBE Box Blur2	Fast Box Blur	32	14.2
	ADBE Gaussian Blur 2	Gaussian Blur	32	13.8
	ADBE Radial Blur	Radial Blur	32	
	ADBE Sharpen	Sharpen	32	13.8
	ADBE Smart Blur	Smart Blur	16	
	ADBE Unsharp Mask2	Unsharp Mask	32	
Channel	ADBE Arithmetic	Arithmetic	8	

continues on next page

Table 1 – continued from previous page

	ADBE Blend	Blend	16	
	ADBE Calculations	Calculations	16	
	CC Composite	CC Composite	16	
	ADBE Channel Combiner	Channel Combiner	8	
	ADBE Compound Arithmetic	Compound Arithmetic	8	
	ADBE Invert	Invert	32	14.1
	ADBE Minimax	Minimax	16	
	ADBE Remove Color Matting	Remove Color Matting	32	
	ADBE Set Channels	Set Channels	16	
	ADBE Set Matte3	Set Matte	32	
	ADBE Shift Channels	Shift Channels	32	
	ADBE Solid Composite	Solid Composite	32	
CINEMA 4D	CINEMA 4D Effect	CINEWARE	32	
Color Correction	ADBE AutoColor	Auto Color	16	
	ADBE AutoContrast	Auto Contrast	16	
	ADBE AutoLevels	Auto Levels	16	
	ADBE Black&White	Black & White	16	
	ADBE Brightness & Contrast 2	Brightness & Contrast	32	14.1
	ADBE Broadcast Colors	Broadcast Colors	8	
	CS Color Neutralizer	CC Color Neutralizer	32	
	CC Color Offset	CC Color Offset	16	
	CS Kernel	CC Kernel	32	
	CC Toner	CC Toner	32	
	ADBE Change Color	Change Color	16	
	ADBE Change To Color	Change to Color	16	
	ADBE CHANNEL MIXER	Channel Mixer	32	
	ADBE Color Balance 2	Color Balance	16	
	ADBE Color Balance (HLS)	Color Balance (HLS)	16	
	ADBE Color Link	Color Link	8	
	ADBE Deflicker	Color Stabilizer	16	
	APC Colorama	Colorama	16	
	ADBE CurvesCustom	Curves	32	
	ADBE Equalize	Equalize	8	
	ADBE Exposure2	Exposure	32	
	ADBE Gamma/Pedestal/Gain2	Gamma/Pedestal/Gain	8	
	ADBE HUE SATURATION	Hue/Saturation	32	14.1
	ADBE Leave Color	Leave Color	8	
	ADBE Easy Levels2	Levels	32	14.2
	ADBE Pro Levels2	Levels (Individual Controls)	32	14.2
	ADBE Lumetri	Lumetri Color	32	13.8
	ADBE PhotoFilterPS	Photo Filter	32	
	ADBE PS Arbitrary Map	PS Arbitrary Map	8	
	ADBE SelectiveColor	Selective Color	16	
	ADBE ShadowHighlight	Shadow/Highlight	16	
	ADBE Tint	Tint	32	14.1
	ADBE Tritone	Tritone	32	
	ADBE Vibrance	Vibrance	16	
Distort	ADBE BEZMESH	Bezier Warp	16	
	ADBE Bulge	Bulge	16	

continues on next page

Table 1 – continued from previous page

	CC Bend It	CC Bend It	16	
	CC Bender	CC Bender	16	
	CC Blobbylize	CC Blobbylize	16	
	CC Flo Motion	CC Flo Motion	32	
	CC Griddler	CC Griddler	32	
	CC Lens	CC Lens	32	
	CC Page Turn	CC Page Turn	16	
	CC Power Pin	CC Power Pin	32	
	CC Ripple Pulse	CC Ripple Pulse	32	
	CC Slant	CC Slant	16	
	CC Smear	CC Smear	32	
	CC Split	CC Split	16	
	CC Split 2	CC Split 2	16	
	CC Tiler	CC Tiler	32	
	ADBE Corner Pin	Corner Pin	32	
	ADBE Upscale	Detail-preserving Upscale	32	
	ADBE Displacement Map	Displacement Map	32	
	ADBE LIQUIFY	Liquify	16	
	ADBE Magnify	Magnify	8	
	ADBE MESH WARP	Mesh Warp	16	
	ADBE Mirror	Mirror	16	
	ADBE Offset	Offset	16	14.2
	ADBE Optics Compensation	Optics Compensation	32	
	ADBE Polar Coordinates	Polar Coordinates	32	
	ADBE RESHAPE	Reshape	16	
	ADBE Ripple	Ripple	16	
	ADBE Rolling Shutter	Rolling Shutter Repair	32	
	ADBE SCHMEAR	Smear	16	
	ADBE Spherize	Spherize	16	
	ADBE Geometry2	Transform	32	15.0
	ADBE Turbulent Displace	Turbulent Displace	32	
	ADBE Twirl	Twirl	32	
	ADBE WRPMESH	Warp	16	
	ADBE SubspaceStabilizer	Warp Stabilizer VFX	32	
	ADBE Wave Warp	Wave Warp	16	
Expression Controls	ADBE Point3D Control	3D Point Control	32	
	ADBE Angle Control	Angle Control	32	
	ADBE Checkbox Control	Checkbox Control	32	
	ADBE Color Control	Color Control	32	
	ADBE Dropdown Control	Dropdown Control	32	
	ADBE Layer Control	Layer Control	32	
	ADBE Point Control	Point Control	32	
	ADBE Slider Control	Slider Control	32	
Generate	ADBE 4ColorGradient	4-Color Gradient	16	
	ADBE Lightning 2	Advanced Lightning	8	
	ADBE AudSpect	Audio Spectrum	32	
	ADBE AudWave	Audio Waveform	32	
	ADBE Laser	Beam	32	
	CC Glue Gun	CC Glue Gun	32	
	CC Light Burst 2.5	CC Light Burst 2.5	32	

continues on next page

Table 1 – continued from previous page

	CC Light Rays	CC Light Rays	32	
	CC Light Sweep	CC Light Sweep	32	
	CS Threads	CC Threads	32	
	ADBE Cell Pattern	Cell Pattern	8	
	ADBE Checkerboard	Checkerboard	8	
	ADBE Circle	Circle	8	
	ADBE ELLIPSE	Ellipse	32	
	ADBE Eyedropper Fill	Eyedropper Fill	8	
	ADBE Fill	Fill	32	
	ADBE Fractal	Fractal	16	
	ADBE Ramp	Gradient Ramp	32	14.2
	ADBE Grid	Grid	8	
	ADBE Lens Flare	Lens Flare	8	
	ADBE Paint Bucket	Paint Bucket	8	
	APC Radio Waves	Radio Waves	8	
	ADBE Scribble Fill	Scribble	8	
	ADBE Stroke	Stroke	8	
	APC Vegas	Vegas	8	
	ADBE Write-on	Write-on	8	
Keying	ADBE Spill2	Advanced Spill Suppressor	32	
	CC Simple Wire Removal	CC Simple Wire Removal	32	
	ADBE Color Difference Key	Color Difference Key	16	
	ADBE Color Range	Color Range	8	
	ADBE Difference Matte2	Difference Matte	16	
	ADBE Extract	Extract	16	
	ADBE ATG Extract	Inner/Outer Key	16	
	ADBE KeyCleaner	Key Cleaner	32	
	Keylight 906	Keylight (1.2)	32	
	ADBE Linear Color Key2	Linear Color Key	32	
Matte	ADBE Matte Choker	Matte Choker	16	
	ISL MochaShapeImporter	mocha shape	32	
	ADBE RefineRBMatte	Refine Hard Matte	32	
	ADBE RefineMatte2	Refine Soft Matte	32	
	ADBE Simple Choker	Simple Choker	32	
Noise & Grain	VISINF Grain Implant	Add Grain	16	
	ADBE Dust & Scratches	Dust & Scratches	16	
	ADBE Fractal Noise	Fractal Noise	32	14.2
	VISINF Grain Duplication	Match Grain	16	
	ADBE Median	Median	16	
	ADBE Noise	Noise	32	
	ADBE Noise Alpha2	Noise Alpha	8	
	ADBE Noise HLS2	Noise HLS	8	
	ADBE Noise HLS Auto2	Noise HLS Auto	8	
	VISINF Grain Removal	Remove Grain	16	
	ADBE AIF Perlin Noise 3D	Turbulent Noise	32	
Obsolete	ADBE Basic 3D	Basic 3D	8	
	ADBE Basic Text2	Basic Text	8	
	ADBE Color Key	Color Key	16	

continues on next page

Table 1 – continued from previous page

	ADBE Fast Blur	Fast Blur (Legacy)	32	
	ADBE Gaussian Blur	Gaussian Blur (Legacy)	32	
	ADBE Lightning	Lightning	8	
	ADBE Luma Key	Luma Key	16	
	ADBE Path Text	Path Text	8	
	ADBE Reduce Interlace Flicker	Reduce Interlace Flicker	32	
	ADBE Spill Suppressor	Spill Suppressor	32	
Perspective	ADBE 3D Tracker	3D Camera Tracker	32	
	ADBE 3D Glasses2	3D Glasses	32	
	ADBE Bevel Alpha	Bevel Alpha	16	
	ADBE Bevel Edges	Bevel Edges	8	
	CC Cylinder	CC Cylinder	16	
	CC Environment	CC Environment	32	
	CC Sphere	CC Sphere	32	
	CC Spotlight	CC Spotlight	16	
	ADBE Drop Shadow	Drop Shadow	32	14.2
	ADBE Radial Shadow	Radial Shadow	8	
Simulation	APC CardDanceCam	Card Dance	8	
	APC Caustics	Caustics	8	
	CC Ball Action	CC Ball Action	16	
	CC Bubbles	CC Bubbles	32	
	CC Drizzle	CC Drizzle	32	
	CC Hair	CC Hair	16	
	CC Mr. Mercury	CC Mr. Mercury	32	
	CC Particle Systems II	CC Particle Systems II	32	
	CC Particle World	CC Particle World	16	
	CC Pixel Polly	CC Pixel Polly	16	
	CSRainfall	CC Rainfall	32	
	CC Scatterize	CC Scatterize	16	
	CSSnowfall	CC Snowfall	32	
	CC Star Burst	CC Star Burst	16	
	APC Foam	Foam	8	
	ADBE Playgnd	Particle Playground	8	
	APC Shatter	Shatter	8	
	APC Wave World	Wave World	8	
Stylize	ADBE Brush Strokes	Brush Strokes	8	
	ADBE Cartoonify	Cartoon	32	
	CS BlockLoad	CC Block Load	32	
	CC Burn Film	CC Burn Film	32	
	CC Glass	CC Glass	16	
	CS HexTile	CC HexTile	32	
	CC Kaleida	CC Kaleida	32	
	CC Mr. Smoothie	CC Mr. Smoothie	16	
	CC Plastic	CC Plastic	16	
	CC RepeTile	CC RepeTile	32	
	CC Threshold	CC Threshold	32	
	CC Threshold RGB	CC Threshold RGB	32	
	CS Vignette	CC Vignette	32	
	ADBE Color Emboss	Color Emboss	16	

continues on next page

Table 1 – continued from previous page

	ADBE Emboss	Emboss	16	
	ADBE Find Edges	Find Edges	8	14.1
	ADBE Glo2	Glow	32	14.1
	ADBE Mosaic	Mosaic	16	
	ADBE Tile	Motion Tile	8	
	ADBE Posterize	Posterize	32	
	ADBE Roughen Edges	Roughen Edges	8	
	ADBE Scatter	Scatter	16	
	ADBE Strobe	Strobe Light	8	
	ADBE Texturize	Texturize	8	
	ADBE Threshold2	Threshold	32	
Synthetic Aperture	SYNTHAP CF Color Finesse 2	SA Color Finesse 3	32	
Text	ADBE Numbers2	Numbers	8	
	ADBE Timecode	Timecode	8	
Time	CC Force Motion Blur	CC Force Motion Blur	32	
	CC Wide Time	CC Wide Time	32	
	ADBE Echo	Echo	32	
	ADBE OFMotionBlur	Pixel Motion Blur	32	
	ADBE Posterize Time	Posterize Time	32	
	ADBE Difference	Time Difference	8	
	ADBE Time Displacement	Time Displacement	16	
	ADBE Timewarp	Timewarp	32	
Transition	ADBE Block Dissolve	Block Dissolve	16	
	APC CardWipeCam	Card Wipe	8	
	CC Glass Wipe	CC Glass Wipe	16	
	CC Grid Wipe	CC Grid Wipe	32	
	CC Image Wipe	CC Image Wipe	16	
	CC Jaws	CC Jaws	32	
	CC Light Wipe	CC Light Wipe	16	
	CS LineSweep	CC Line Sweep	32	
	CC Radial ScaleWipe	CC Radial ScaleWipe	16	
	CC Scale Wipe	CC Scale Wipe	32	
	CC Twister	CC Twister	16	
	CC WarpoMatic	CC WarpoMatic	16	
	ADBE Gradient Wipe	Gradient Wipe	16	
	ADBE IRIS_WIPE	Iris Wipe	32	
	ADBE Linear Wipe	Linear Wipe	32	
	ADBE Radial Wipe	Radial Wipe	32	
	ADBE Venetian Blinds	Venetian Blinds	32	
Utility	ADBE Apply Color LUT2	Apply Color LUT	32	
	CC Overbrights	CC Overbrights	32	
	ADBE Cineon Converter2	Cineon Converter	32	
	ADBE ProfileToProfile	Color Profile Converter	32	
	ADBE GROW BOUNDS	Grow Bounds	32	
	ADBE Compander	HDR Compander	32	
	ADBE HDR ToneMap	HDR Highlight Compression	32	

continues on next page

Table 1 – continued from previous page

_Obsolete	ADBE Paint	Paint		
	ADBE Samurai	Roto Brush & Refine Edge		
	ADBE FreePin3	Puppet		
	ADBE RefineMatte	Refine Matte		
	ADBE 3D Glasses	3D Glasses (Obsolete)		
	ADBE Alpha Levels2	Alpha Levels		
	ADBE Alpha Levels3	Alpha Levels		
	ADBE Apply Color LUT	Apply Color LUT		
	ADBE Brightness & Contrast	Brightness & Contrast		
	ADBE Box Blur	Box Blur		
	ADBE Cineon Converter	Cineon Converter		
	ADBE Color Balance	Color Balance		
	CC PS Classic	CC PS Classic (obsolete)		
	CC PS LE Classic	CC PS LE Classic (obsolete)		
	CC Rain	CC Rain		
	CC Snow	CC Snow		
	CC Time Blend	CC Time Blend		
	CC Time Blend FX	CC Time Blend FX		
	ADBE Exposure	Exposure		
	ADBE Easy Levels	Levels		
	ADBE Pro Levels	Levels (Individual Controls)		
	ADBE Noise Alpha	Noise Alpha		
	ADBE Noise HLS	Noise HLS		
	ADBE Noise HLS Auto	Noise HLS Auto		
	ADBE PSL Bevel Emboss	Photoshop Bevel And Emboss		
	ADBE PSL Drop Shadow	Photoshop Drop Shadow		
	ADBE PSL Inner Glow	Photoshop Inner Glow		
	ADBE PSL Inner Shadow	Photoshop Inner Shadow		
	ADBE PSL Outer Glow	Photoshop Outer Glow		
	ADBE PSL Solid Fill	Photoshop Solid Fill		
	ADBE Photo Filter	Photo Filter		
	ADBE Set Matte2	Set Matte		
	ADBE Three-Way Color Corrector	Three-Way Color Corrector		
	ADBE Threshold	Threshold		
	ADBE Geometry	Transform		
	ADBE Unsharp Mask	Unsharp Mask		
	ADBE Vector Paint	Vector Paint		