

---

# Premiere Pro Scripting Guide

*Release 1.0*

Feb 02, 2019



<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Overview</b>	<b>3</b>
2.1	Example code . . . . .	3
2.2	Development and debugging tools . . . . .	3
<b>3</b>	<b>How to Execute ExtendScript in Premiere Pro</b>	<b>5</b>
<b>4</b>	<b>Application object</b>	<b>7</b>
4.1	Attributes . . . . .	7
4.2	Methods . . . . .	10
<b>5</b>	<b>Project object</b>	<b>15</b>
5.1	Attributes . . . . .	15
5.2	Methods . . . . .	17
<b>6</b>	<b>Project Item object</b>	<b>25</b>
6.1	Attributes . . . . .	25
6.2	Methods . . . . .	27
<b>7</b>	<b>Sequence object</b>	<b>41</b>
7.1	Attributes . . . . .	41
7.2	Methods . . . . .	44
<b>8</b>	<b>Track object</b>	<b>53</b>
8.1	Attributes . . . . .	53
8.2	Methods . . . . .	54
<b>9</b>	<b>Track Item object</b>	<b>57</b>
9.1	Attributes . . . . .	57
9.2	Methods . . . . .	60
<b>10</b>	<b>Component object</b>	<b>63</b>
10.1	Attributes . . . . .	63
<b>11</b>	<b>Component Parameter object</b>	<b>65</b>
11.1	Attributes . . . . .	65
11.2	Methods . . . . .	65

<b>12 Anywhere object</b>	<b>73</b>
12.1 Attributes . . . . .	73
12.2 Methods . . . . .	73
<b>13 Encoder object</b>	<b>77</b>
13.1 Attributes . . . . .	77
13.2 Methods . . . . .	77
<b>14 Marker object</b>	<b>81</b>
14.1 Attributes . . . . .	81
14.2 Methods . . . . .	83
<b>15 Source object</b>	<b>87</b>
15.1 Attributes . . . . .	87
15.2 Methods . . . . .	87

# CHAPTER 1

---

## Introduction

---

This reference is to serve as a public compendium of information about the methods and members available via the API.

Premiere Pro provides an ExtendScript-based API, allowing for broad control of the entire application. ExtendScript can access and manipulation of most project elements, including metadata, exporting and rendering options.

Adobe wants your integration to succeed; please don't hesitate to [contact us](#) with questions, problems, or feature requests.



Premiere Pro provides an ExtendScript API, allowing for the access and manipulation of most project elements, including metadata, exporting and rendering options.

---

**Note:** This document does not teach ExtendScript, ExtendScript Toolkit debugging, or other development techniques. It focuses on the Premiere Pro ExtendScript API and the execution context for scripts.

---

While initially incomplete and intended only for internal testing, the Premiere Pro ExtendScript API has been growing steadily for many years. As of 12.1.1 (the current release, as of this writing), the API offers thorough access to (and, often, control over) all project elements, as well as application settings.

## 2.1 Example code

The PProPanel sample exercises Premiere Pro's ExtendScript API: <https://github.com/Adobe-CEP/Samples/tree/master/PProPanel>.

## 2.2 Development and debugging tools

The ExtendScript Toolkit (ESTK) remains the best (and only) ExtendScript debugger available.

<http://www.adobe.com/products/extendscript-toolkit.html>

ESTK can be used to execute stand-alone scripts in Premiere Pro, and debug ExtendScript invoked from within CEP panels. Commonly, developers test individual functions in ESTK, then debug them 'in context' running from within a CEP panel.





---

### How to Execute ExtendScript in Premiere Pro

---

Executing scripts from within CEP panels is the recommended approach. It is also possible to invoke scripts from within ExtendScript Toolkit (ESTK), but that's not a user-friendly approach for most workflows. Also, with additional configuration work, it's possible to pass scripts to Premiere Pro on a command line. This is not recommended, as behavior varies across platforms, and preliminary configuration is necessary before such execution is enabled.

Use ESTK for development and debugging; for deployed workflows, stick with panels whenever possible.



---

## Application object

---

app

### Description

Provides access to objects and application settings within the Premiere Pro application. The single global object is always available by its name, **app**.

Attributes of the Application object provide access to specific objects within Premiere Pro. Methods of the Application object can create, open and close projects, access other APIs available from the PPro DOM (Anywhere, Team Projects, Encoder), and quit Premiere Pro.

---

## 4.1 Attributes

### 4.1.1 version

app.version

### Description

The version of Premiere Pro, providing the API.

### Type

Floating point; read-only.

---

### 4.1.2 buildNumber

`app.buildNumber`

#### Description

The number of the build of Premiere Pro being run.

#### Type

Integer; read-only.

---

### 4.1.3 getPProPrefPath

`app.getPProPrefPath`

#### Description

The path containing the currently active “Adobe Premiere Pro Prefs” file.

#### Type

String; read-only.

---

### 4.1.4 getPProSysPrefPath

`app.getPProSysPrefPath`

#### Description

The path containing Premiere Pro’s active configuration files, not specific to a given user.

#### Type

String; read-only.

---

### 4.1.5 project

`app.project`

#### Description

The currently active project. Before Premiere Pro 12.0, this pointed at the *only* project; after 12.0, this points to the currently active project.

#### Type

String; read-only.

---

### 4.1.6 projects

`app.projects`

#### **Description**

An Array referencing all open projects.

#### **Type**

Array (of Project objects); read-only.

---

### 4.1.7 anywhere

`app.anywhere`

#### **Description**

An Anywhere object, providing access to connected Anywhere servers, through the Premiere Pro DOM.

**Type** Anywhere object; read-only. —

### 4.1.8 Encoder

`app.Encoder`

#### **Description**

An Encoder object, providing access to Adobe Media Encoder (as used by Premiere Pro), through the Premiere Pro DOM.

#### **Type**

Encoder object; read-only.

---

### 4.1.9 projectManager

`app.projectManager`

#### **Description**

A projectManager object, providing access to project management functions within Premiere Pro.

#### **Type**

projectManager object; read-only.

---

### 4.1.10 userGuid

`app.userGuid`

#### Description

A unique identifier for the currently logged-in Creative Cloud user.

#### Type

userGuid object; read-only.

---

### 4.1.11 properties

`app.properties`

#### Description

The properties object provides methods to access and modify preference values.

#### Type

properties object; read-only.

---

### 4.1.12 sourceMonitor

`app.sourceMonitor`

#### Description

Provides access to Source monitor.

#### Type

sourceMonitor object; read-only.

---

## 4.2 Methods

### 4.2.1 isDocumentOpen()

`app.isDocumentOpen()`

#### Description

Determines whether there are any projects currently open.

#### Parameters

None.

#### Returns

Returns **true** if at least 1 project is open; otherwise **false**.

---

### 4.2.2 isDocument(path)

```
app.isDocument (path)
```

#### Description

Determines whether the file at path can be opened as a Premiere Pro project.

#### Parameters

None.

#### Returns

Returns **true** if file is openeable.

---

### 4.2.3 openDocument(path)

```
app.openDocument (path)
```

#### Description

Opens the file at the specified path, as a Premiere Pro project.

#### Parameters

path

#### Returns

Returns **true** if file was successfully opened.

---

### 4.2.4 openFCPXML(path, projPath)

```
app.openFCPXML (path, projPath)
```

#### Description

Opens an FCP XML file at path, as a Premiere Pro project (specified in projPath).

#### Parameters

path, projPath.

#### Returns

Returns **true** if file was successfully opened as a Premiere Pro project.

---

### 4.2.5 quit()

```
app.quit()
```

#### Description

Quits Premiere Pro; user will be prompted to save any changes to project.

#### Parameters

None.

#### Returns

Nothing.

---

### 4.2.6 trace()

```
app.trace()
```

#### Description

Writes a string to Premiere Pro's debug console.

#### Parameters

None.

#### Returns

Nothing.

---

### 4.2.7 setSDKEventMessage()

```
app.setSDKEventMessage(message, decorator)
```

#### Description

Writes a string to Premiere Pro's Events panel.

#### Parameters

message is a string; decorator can be either 'info', 'warning' or 'error'.

#### Returns

Returns 'true' if successful.

---



### 4.2.8 setScratchDiskPath()

```
app.setScratchDiskPath(path, whichScratchValueToSet)
```

#### Description

Specifies the path to be used for one of Premiere Pro's scratch disk paths.

#### Parameters

path	The new path to be used.
whichScratchValueToSet	Must be one of the following: FirstAudioCaptureFolder FirstVideoCaptureFolder FirstAudioPreviewFolder FirstAutoSaveFolder FirstCCLibrariesFolder

#### Returns

Returns 'true' if successful.

### 4.2.9 enableQE()

```
app.enableQE()
```

#### Description

Enables Premiere Pro's QE DOM.

#### Parameters

None.

#### Returns

Returns true if QE DOM was enabled.

### 4.2.10 setExtensionPersistent(ExtensionID, persist)

```
app.setExtensionPersistent(ExtensionID, persist)
```

#### Description

Controls whether the extension with the given ExtensionID remains persistent in memory, throughout the Premiere Pro session.

#### Parameters

extensionID	Which extension to modify.
persist	Pass 1 to keep extension in memory, 0 to allow unloading.

Returns Nothing. —

### 4.2.11 `getEnableProxies()`

```
app.getEnableProxies()
```

#### **Description**

Determines whether proxy usage is currently enabled.

#### **Parameters**

None.

#### **Returns**

Returns 1 if proxies are enabled, 0 if they are not.

---

### 4.2.12 `setEnableProxies(enabled)`

```
app.setEnableProxies(enabled)
```

#### **Description**

Determines whether proxy usage is currently enabled.

#### **Parameters**

<code>enabled</code>	1 turns proxies on, 0 turns them off.
----------------------	---------------------------------------

#### **Returns**

Returns 1 if proxy enablement was changed.

`project`

### **Description**

Represents a Premiere Pro project. As of Premiere Pro 12.0, multiple projects may be open at the same time.

---

## **5.1 Attributes**

### **5.1.1 documentID**

`project.documentID`

#### **Description**

A read-only unique identifier for this project.

#### **Type**

GUID; read-only.

---

### **5.1.2 name**

`project.name`

#### **Description**

The name of the project.

#### **Type**

String; read-only.

---

### 5.1.3 path

`project.path`

#### Description

The file path of the project.

#### Type

String; read-only.

---

### 5.1.4 rootItem

`project.rootItem`

#### Description

A `projectItem` representing the “root” of the project.

#### Type

A **projectItem**; this will always be of type `ProjectItemType_BIN`.

---

### 5.1.5 sequences

`project.sequences`

#### Description

The sequences within the project.

#### Type

An array of sequence objects.

---

### 5.1.6 activeSequence

`project.activeSequence`

#### Description

The currently active sequence, within the project.

#### Type

a sequence object, or 0 if no sequence is currently active.

---

## 5.2 Methods

### 5.2.1 openSequence()

```
project.openSequence(sequenceID)
```

**Description**

Makes the sequence with the provided sequence ID, active. This will open the sequence in the Timeline panel.

**Parameters**

A valid `sequenceID`.

**Returns**

Returns **true** if successful, **false** if not.

---

### 5.2.2 importFiles()

```
project.importFiles(arrayOfFilePathsToImport, suppressUI, targetBin,
importAsNumberedStills)
```

**Description**

Imports media from the specified file paths.

**Parameters**

An array of file paths to be imported, a `boolean` indicating whether warning dialogs should be suppressed, a `projectItem` object for the bin into which the files should be imported, and a `boolean` indicating whether the file paths should be interpreted as a sequence of numbered stills.

**Returns**

Returns **true** if successful, **false** if not.

---

### 5.2.3 importSequences()

```
project.importSequences(pathOfContainingProject, arrayOfSequenceIDs)
```

**Description**

Imports an array of sequences (with specified `sequenceIDs`), from the specified project, into the current project.

**Parameters**

*String* containing the full path to the containing project file, and an *Array* of `sequenceIDs`.

**Returns**

Returns **0** if successful.

---

## 5.2.4 importAECOMPs()

```
project.importAECOMPs (pathOfContainingProject, arrayOfCompNames,  
optionalTargetBin)
```

### Description

Imports specified Compositions (by name) from the containing After Effects .aep project file. You can specify a target bin within the containing project; otherwise, the Compositions will appear in the most recently targeted bin, within this project.

### Parameters

*String* containing the full path to the containing project file, and an *Array* of sequenceIDs.

*Array* of names of Compositions within the specified project, to be imported.

*projectItem* referencing the destination bin for this import.

### Returns

Returns **0** if successful.

---

## 5.2.5 importAllAECOMPs()

```
project.importAllAECOMPs (pathOfContainingProject, optionalTargetBin)
```

### Description

Imports specified Compositions (by name) from the containing After Effects .aep project file. You can specify a target bin within the containing project; otherwise, the Compositions will appear in the most recently targeted bin, within this project.

### Parameters

*String* containing the full path to the containing project file.

*projectItem* referencing the destination bin for this import.

### Returns

Returns **0** if successful.

---

## 5.2.6 createNewSequence()

```
project.createNewSequence (sequenceName, sequenceID)
```

### Description

Creates a new sequence with the specified ID.

### Parameters

*String* name of sequence.

*GUID* uniquely identifying this sequence.

### Returns

Returns a **Sequence** object if creation was successful, or **0** if unsuccessful.

---

### 5.2.7 deleteSequence()

```
project.deleteSequence(sequenceToDelete)
```

**Description**

Deletes the specified sequence from the project.

**Parameters**

The **Sequence** to delete.

**Returns**

Returns 0 if successful.

---

### 5.2.8 exportFinalCutProXML()

```
project.exportFinalCutProXML(outputPath, suppressUI)
```

**Description**

Exports an FCP XML representation of the entire project, to the specified output path.

**Parameters**

Full output path of .xml file, as a *String*.

The suppressUI param is an *Int*; if **1**, no warnings or alerts will be shown, during the export.

**Returns**

Returns 0 if successful.

---

### 5.2.9 exportTimeline()

```
project.exportTimeline(exportControllerName)
```

**Description**

Exports the currently active sequence, using an Export Controller plug-in with the specified name.

**Parameters**

A **String** containing the name of the Export Controller plug-in to be used. To use the Premiere Pro SDK example Export Controller, the value would be “SDK Export Controller”.

**Returns**

Returns **0** if successful, or an error code if not.

---

### 5.2.10 project.exportOMF()

```
project.exportOMF(sequence, outputPath, omfTitle, sampleRate, bitsPerSample, audioEncapsulated, audioFileFormat, trimAudioFiles, handleFrames, includePan)
```

#### Description

Exports an OMF file of the specified sequence, using the specified settings.

#### Parameters

sequence	Specifies the sequence to be output.
filePath	Complete output path for .omf file.
omfTitle	<b>String</b> with which to title the OMF.
sampleRate	Specifies the sample rate of output audio.
bitsPerSample	Specifies the bits per sample of audio output.
audioEncapsulated	If <b>1</b> , audio is embedded, if <b>0</b> , external.
audioFileFormat	<b>0</b> is AIFF, <b>1</b> is WAV.
trimAudioFiles	<b>1</b> means yes, trim audio files.
handleFrames	Number of handle frames (from 0 to 1000).
includePan	<b>1</b> means include pan info; <b>0</b> means don't.

#### Returns

Returns **0** if successful.

---

### 5.2.11 exportAAF()

```
project.exportAAF(sequenceToExport, outputPath, mixdownVideo, explodeToMono, sampleRate, bitsPerSample, embedAudio, audioFileFormat, trimSources, handleFrames, presetPath, renderAudioEffects, includeClipCopies, preserveParentFolder)
```

#### Description

Exports an AAF file of the specified sequence, using the specified settings.

#### Parameters

sequence	Specifies the sequence to be output.
filePath	Complete output path for .aaf file.
mixdownVideo	If <b>1</b> , render video before export.
explodeToMono	If <b>1</b> , breaks out stereo tracks to mono.
sampleRate	Specifies the sample rate of output audio.
bitsPerSample	Specifies the bits per sample of audio output.
embedAudio	If <b>1</b> , audio is embedded, if <b>0</b> , external.
audioFileFormat	<b>0</b> is AIFF, <b>1</b> is WAV.
trimSources	If <b>1</b> , trim audio files before export.
handleFrames	Number of handle frames (from 0 to 1000).
presetPath	Complete path to Export preset (.epr file).
renderAudioEffects	If <b>1</b> , render audio effects before export.
includeClipCopies	If <b>1</b> , include each copy of a clip.
preserveParentFolder	If <b>1</b> , preserves the parent folder, in output.



**Returns**

Returns **0** if successful.

---

**5.2.12 saveAs()**

```
project.saveAs(pathToNewProject)
```

**Description**

Exports the current project to a new unique file path, opens the project from the new location, and closes the previously-opened (and identical) project.

**Parameters**

A **String** specifying the new path.

**Returns**

Returns **0** if successful, or an error code if not.

---

**5.2.13 save()**

```
project.save()
```

**Description**

Saves the project, at its current path.

**Parameters**

None.

**Returns**

Returns **0** if successful.

---

**5.2.14 pauseGrowing()**

```
project.pauseGrowing(pausedOrNot)
```

**Description**

Pauses (and resumes) growing file capture.

**Parameters**

An **int**; if 1, growing files are enabled.

**Returns**

Returns **0** if successful.

---

### 5.2.15 closeDocument()

```
project.closeDocument(saveFirst, promptIfDirty)
```

#### Description

Closes this project.

#### Parameters

Two **ints**; If **saveFirst** is 1, the project will be saved before closing. If **promptIfDirty** is 1, the user will be asked whether they want to save changes first.

#### Returns

Returns **0** if successful.

---

### 5.2.16 addPropertyToProjectMetadataSchema()

```
project.addPropertyToProjectMetadataSchema(propertyName, propertyLabel, propertyType)
```

#### Description

Adds a new field of the specified type to Premiere Pro's private project metadata schema.

#### Parameters

propertyName	<b>String</b> , Name of property to be added.
propertyLabel	<b>String</b> , Label of property to be added.
propertyType	<b>Must be one of the following:</b> <ul style="list-style-type: none"><li>• 0 Integer</li><li>• 1 Real</li><li>• 2 String</li><li>• 3 Boolean</li></ul>

#### Returns

Returns **0** if successful.

---

### 5.2.17 getInsertionBin()

```
project.getInsertionBin()
```

#### Description

Returns a **projectItem** referencing the bin into which import will occur.

#### Parameters

None.

#### Returns

Returns a **projectItem** if successful, **0** if not.

---

### 5.2.18 `getProjectPanelMetadata()`

```
project.getProjectPanelMetadata()
```

#### Description

Returns the current layout of the Project panel.

#### Parameters

None.

#### Returns

Returns a **String** representing the current Project panel layout, or **0** if unsuccessful.

### 5.2.19 `setProjectPanelMetadata()`

```
project.setProjectPanelMetadata(updatedLayoutAsString)
```

#### Description

Returns the current layout of the Project panel.

#### Parameters

**updatedLayoutAsString** represents the desired Project panel layout. Note: The only known method for generating a valid layout string, is setting the Project panel as desired then using *project.getProjectPanelMetadata*.

#### Returns

Returns **0** if unsuccessful.

### 5.2.20 `setScratchDiskPath()`

```
project.setScratchDiskPath(newPath, whichScratchDiskPath)
```

#### Description

Changes the specified scratch disk path to a new path.

#### Parameters

<code>newPath</code>	New path value.
<code>whichScratchDiskPath</code>	<p><b>Must be one of the following:</b></p> <ul style="list-style-type: none"> <li>• FirstVideoCaptureFolder</li> <li>• FirstAudioPreviewFolder</li> <li>• FirstAutoSaveFolder</li> <li>• FirstCCLibrariesFolder</li> <li>• FirstVideoCaptureFolder</li> <li>• FirstAudioCaptureFolder</li> </ul>

#### Returns

Returns **0** if unsuccessful.

### 5.2.21 consolidateDuplicates()

```
project.consolidateDuplicates()
```

#### **Description**

Invokes Premiere Pro's "Consolidate Duplicate Footage" functionality, as available from the UI.

#### **Parameters**

None.

#### **Returns**

Returns **0** if successful.

---

## Project Item object

---

`projectItem`

### Description

Each item in a given Premiere Pro project is a **projectItem**. The root item of each project is also, itself, a **projectItem**.

---

## 6.1 Attributes

### 6.1.1 children

`projectItem.children`

### Description

A read-only array of project items, which are themselves contained within the specified project item.

### Type

Array; read-only.

---

### 6.1.2 name

`projectItem.name`

### Description

The name of the project item.

### Type

String; read/write.

---

### 6.1.3 treePath

`projectItem.treePath`

#### Description

The path within the project, to the current location of the project item. For example, an .mxf file located in a bin named “MXF”, within a bin named “Media”, would have a tree path of

`\ProjectName.prproj\Media\MXF\filename.mxf`

#### Type

String; read-only.

---

### 6.1.4 type

`projectItem.type`

#### Description

The type of the project Item. Value will be one of the following: **CLIP**, **BIN**, **ROOT**, or **FILE**.

#### Type

Enumeration; read-only.

---

### 6.1.5 nodeID

`projectItem.nodeID`

#### Description

A unique ID assigned to the project item, upon its addition to the project.

**NOTE:** This allows for differentiation between two different project items, associated with the same source media.

#### Type

String; read-only.

---

### 6.1.6 videoComponents

`projectItem.videoComponents`

#### Description

An array of video components associated with the ‘Master Clip’ of this project item.

#### Type

Array of video components. While the parameters associated with these components are read/write, this array of components is, itself, read-only.

---

### 6.1.7 teamProjectsAssetId

```
projectItem.teamProjectsAssetId
```

**Description**

The Team Projects Asset ID of the project item.

**Type** String; read-only.

---

## 6.2 Methods

### 6.2.1 createSmartBin()

```
projectItem.createSmartBin(String nameOfNewBin, String queryString)
```

**Description**

Creates a smart bin (also known as a search bin), within the project item. Only works within project items which are bins.

**Parameters**

Name of new bin. Query string for search.

**Returns**

Returns **0** if creation if smart bin was successful.

---

### 6.2.2 createBin()

```
projectItem.createBin(String nameOfNewBin)
```

**Description**

Creates an empty bin, within the project item. Only works within project items which are bins.

**Parameters**

Name of new bin.

**Returns**

Returns **0** if creation of bin was successful.

---

### 6.2.3 renameBin()

```
projectItem.renameBin(newName)
```

**Description**

Changes name of bin. Only works on project items which are bins.

**Parameters**

New bin name.

**Returns**

Returns **0** if renaming bin was successful.

---

### 6.2.4 moveBin()

```
projectItem.moveBin(newParentBinProjectItem)
```

**Description**

Moves the projectItem into a new parent bin.

**Parameters**

None.

**Returns**

Returns **0** if move was successful.

---

### 6.2.5 deleteBin()

```
projectItem.deleteBin()
```

**Description**

Deletes a bin, **AND ALL ITS CONTENTS**, from the project. Only works on project items which are bins.

**Parameters**

None.

**Returns**

Returns **0** if deletion was successful.

---

### 6.2.6 getStartTime()

```
projectItem.getStartTime()
```

**Description**

Returns a Time object, representing the start time of the project item.

**Parameters**

None.

**Returns**

A Time object.

---



### 6.2.7 getXMPMetadata()

```
projectItem.getXMPMetadata ()
```

#### Description

Retrieves the XMP metadata associated with the project item, as a String.

#### Parameters

None.

#### Returns

A String containing all XMP metadata, serialized.

---

### 6.2.8 setXMPMetadata()

```
projectItem.setXMPMetadata (newXMPAsString)
```

#### Description

Sets the XMP metadata associated with the project item.

#### Parameters

A String representing the new, serialized XMP metadata.

#### Returns

Returns 0 if update was successful.

---

### 6.2.9 getProjectMetadata()

```
projectItem.getProjectMetadata ()
```

#### Description

Retrieves Premiere Pro's private project metadata associated with the project item, as a String. **NOTE** While this data is also valid XMP, it is distinct from the XMP metadata associated with the media.

#### Parameters

None.

#### Returns

A String containing all Premiere Pro private project metadata, serialized.

---

### 6.2.10 setProjectMetadata()

```
projectItem.setProjectMetadata (String newPrivateProjectMetadata,  
arrayOfUpdatedFields)
```

#### Description

Sets the private project metadata associated with the project item.

#### Parameters

A String representing the new, serialized private project metadata, and an array containing the names of the fields to be updated.

#### Returns

Returns 0 if update was successful.

---

### 6.2.11 getMarkers()

```
projectItem.getMarkers ()
```

#### Description

Retrieves the markers associated with this project item.

#### Parameters

None.

#### Returns

An array of markers associated with the project item, or 0 if there are no markers.

---

### 6.2.12 refreshMedia()

```
projectItem.refreshMedia ()
```

#### Description

Forces Premiere Pro to update its representation of the media associated with the project item. If the media was previously off-line, this can cause it to become online (if previously missing media has become available).

#### Parameters

None.

#### Returns

An array of markers associated with the project item, or 0 if there are no markers.

---

### 6.2.13 getMediaPath()

```
projectItem.getMediaPath()
```

#### Description

Returns the path associated with the project item's media, as a String. **NOTE:** This only works for atomic media; this call cannot provide meaningful paths for media which has no actual path (which will be the case for any media generated by synthetic importers, like Premiere Pro's own Universal Counting Leader).

#### Parameters

None.

#### Returns

A String containing the path to the media associate with the project item.

---

### 6.2.14 canChangeMediaPath()

```
projectItem.canChangeMediaPath()
```

#### Description

Returns **true** if Premiere Pro can change the path, associated with this project item; otherwise, returns **false**.

#### Parameters

None.

#### Returns

Boolean; **true** if media can be replaced, **false** if not.

---

### 6.2.15 changeMediaPath()

```
projectItem.changeMediaPath(String newPath)
```

#### Description

Updates the project item to point to a new media path.

#### Parameters

A String, representing the new path.

#### Returns

Returns **0** if replacement was successful.

---

### 6.2.16 select()

```
projectItem.select()
```

#### Description

Sets the project item (which must be a bin), as the target for subsequent imports into the project.

#### Parameters

None.

#### Returns

Returns **0** if the project item has successfully been made the target, for subsequent imports.

---

### 6.2.17 setOverridePixelAspectRatio()

```
projectItem.setOverridePixelAspectRatio(int numerator, int denominator)
```

#### Description

Sets the pixel aspect ratio for the project item.

#### Parameters

Integers representing the new numerator and denominator.

#### Returns

Returns **0** if the aspect ratio has successfully been changed.

---

### 6.2.18 setOverrideFrameRate()

```
projectItem.setOverrideFrameRate(float newFrameRate)
```

#### Description

Sets the frame rate of the project item.

#### Parameters

**Float** representing the new frame rate.

#### Returns

Returns **0** if the frame rate has successfully been changed.

---

### 6.2.19 setScaleToFrameSize()

```
projectItem.setScaleToFrameSize()
```

#### Description

Turns on scaling to frame size, for when media from this project item is inserted into a sequence.

#### Parameters

None.

#### Returns

Undefined return value.

### 6.2.20 createSubClip()

```
projectItem.createSubClip(subclipName, startTime, endTime, hasHardBoundaries,
takeAudio, takeVideo)
```

#### Description

Creates a new project item for a sub-clip of the existing project item.

#### Parameters

subclipName	Name of new subclip.
startTime	Start time of subclip, in <b>Ticks</b> .
endTime	End time of subclip, in <b>Ticks</b> .
hasHardBoundaries	0 or 1; if 1, the user cannot extend in and out.
takeVideo	0 or 1; if 1, use video from source.
takeAudio	0 or 1; if 1, use video from source.

#### Returns

Returns a project item representing the new subclip, or 0 if creation failed.

### 6.2.21 findItemsMatchingMediaPath()

```
projectItem.findItemsMatchingMediaPath(pathToMatch, ignoreSubClips)
```

#### Description

Returns an array of project items, all of which reference the same media path.

#### Parameters

pathToMatch	Path to match, as <b>String</b> .
ignoreSubClips	0 or 1; if 1, no subclips will be returned.

#### Returns

Returns an array of project items, or **0** if no project items matching the `matchPath` were found.

### 6.2.22 canProxy()

```
projectItem.canProxy()
```

#### Description

Indicates whether it's possible to attach a proxy, to this project item.

#### Parameters

None.

#### Returns

Returns **true** if the project item permits a proxy to be attached; **false** if not.

---

### 6.2.23 hasProxy()

```
projectItem.hasProxy()
```

#### Description

Indicates whether a proxy has already been attached, to the project item.

#### Parameters

None.

#### Returns

Returns **true** if the project item has a proxy attached; **false** if not.

---

### 6.2.24 getProxyPath()

```
projectItem.getProxyPath()
```

#### Description

Retrieves the path to the proxy media associated with this project item.

#### Parameters

None.

#### Returns

Returns the path (as **String**) to the proxy media associated with the proxy item, or **0** if none is found.

---

### 6.2.25 attachProxy()

```
projectItem.attachProxy(String newMediaPath, int isHiRes)
```

**Description**

Attaches the media at `newMediaPath` to the project item, as either hi-res or proxy media.

**Parameters**

The path the the newly-assigned media (as `String`), and an **int** indicating whether the new media should be attached as the proxy (**0**) or high resolution (**1**) media.

**Returns**

Returns **0** if successful.

---

### 6.2.26 IsSequence()

```
projectItem.IsSequence()
```

**Description**

Indicates whether the project item refers to a sequence.

**Parameters**

None.

**Returns**

Returns `true` if the project item is a sequence, `false` if it isn't.

---

### 6.2.27 setStartTime()

```
projectItem.setStartTime(timeInTicks)
```

**Description**

Assigns a new start time to the project item

**Parameters**

New starting time, represented in ticks.

**Returns**

Returns **0** if successful.

---

### 6.2.28 getOutPoint()

```
projectItem.getOutPoint(mediaType)
```

#### Description

Retrieves the current out point for specified media type.

#### Parameters

`mediaType` is an `int`; pass 1 for video only, or 2 for audio only. If no `mediaType` is passed, function gets the out point for all media.

#### Returns

Returns a `Time` object.

---

### 6.2.29 setOutPoint()

```
projectItem.setOutPoint(timeInTicks, mediaType)
```

#### Description

Sets the out point to `timeInTicks`, for specified media types. `mediaType` defaults to all; pass 1 for video only, or 2 for audio only.

#### Parameters

A `Time` object, and an `int`; pass 1 for video only, or 2 for audio only. If no `mediaType` is passed, function sets the out point for all media.

#### Returns

Returns 0 if successful.

---

### 6.2.30 clearOutPoint()

```
projectItem.clearOutPoint()
```

#### Description

Clears any assigned out point; the project item will then start at `startTime`.

#### Parameters

None

#### Returns

Returns 0 if successful.

---



### 6.2.31 getColorLabel()

```
projectItem.getColorLabel()
```

#### Description

Retrieves the project item's color label.

#### Parameters

None.

#### Returns

labelColor	<ul style="list-style-type: none"> <li>• 0 = Violet</li> <li>• 1 = Iris</li> <li>• 2 = Caribbean</li> <li>• 3 = Lavender</li> <li>• 4 = Cerulean</li> <li>• 5 = Forest</li> <li>• 6 = Rose</li> <li>• 7 = Mango</li> <li>• 8 = Purple</li> <li>• 9 = Blue</li> <li>• 10 = Teal</li> <li>• 11 = Magenta</li> <li>• 12 = Tan</li> <li>• 13 = Green</li> <li>• 14 = Brown</li> <li>• 15 = Yellow</li> </ul>
------------	--

### 6.2.32 setColorLabel()

```
projectItem.setColorLabel(newLabelColor)
```

#### Description

Sets the project item's color label.

#### Parameters

New label color; see *projectItem.getColorLabel*.

#### Returns

0 if successful.

### 6.2.33 getFootageInterpretation()

```
projectItem.getFootageInterpretation()
```

#### Description

Returns a structure describing the current interpretation of the projectItem.

**Parameters**

None.

**Returns**

A footage interpretation structure, or 0 if unsuccessful.

alphaUsage	<p><b>Alpha, will be one of the following:</b></p> <ul style="list-style-type: none"> <li>• 0 ALPHACHANNEL_NONE</li> <li>• 1 ALPHACHANNEL_STRAIGHT</li> <li>• 2 ALPHACHANNEL_PREMULTIPLIED</li> <li>• 3 ALPHACHANNEL_IGNORE</li> </ul>
fieldType	<p><b>Field type, one of the following:</b></p> <ul style="list-style-type: none"> <li>• -1 FIELDTYPE_DEFAULT</li> <li>• 0 FIELDTYPE_PROGRESSIVE</li> <li>• 1 ALPHACHANNEL_UPPERFIRST</li> <li>• 2 ALPHACHANNEL_LOWERFIRST</li> </ul>
ignoreAlpha	true or false.
invertAlpha	true or false.
frameRate	Frame rate as floating point value.
pixelAspectRatio	Pixel aspect ratio as floating point value.
removePulldown	true or false.
vrConformProjectionType	<p><b>The projection type in use, for VR footage. One of these:</b></p> <ul style="list-style-type: none"> <li>• 0 VR_CONFORM_PROJECTION_NONE</li> <li>• 1 VR_CONFORM_PROJECTION_EQUIRECTANGULAR</li> </ul>
vrLayoutType	<p><b>The layout of footage in use, for VR. One of these:</b></p> <ul style="list-style-type: none"> <li>• 0 VR_LAYOUT_MONOSCOPIC</li> <li>• 1 VR_LAYOUT_STEREO_OVER_UNDER</li> <li>• 2 VR_LAYOUT_STEREO_SIDE_BY_SIDE</li> </ul>
vrHorizontalView	The horizontal view in use, for VR footage.
vrVerticalView	The vertical view in use, for VR footage.

**6.2.34 setFootageInterpretation()**

projectItem.setFootageInterpretation(newInterpretation)

**Description**

Returns a structure describing the current interpretation of the projectItem.

**Parameters**

A footage interpretation structure.

**Returns**

0 if successful.

---

**6.2.35 isOffline()**

```
projectItem.isOffline()
```

**Description**

Returns a Boolean indicating whether the project item is offline.

**Parameters**

None.

**Returns**

Boolean, `true` if offline.

---

**6.2.36 setOffline()**

```
projectItem.setOffline()
```

**Description**

Makes the project item offline.

**Parameters**

None.

**Returns**

`true` if successful.



---

## Sequence object

---

Sequence

### Description

The **Sequence** object represents sequences of media (a.k.a. “timelines”), in Premiere Pro.

---

## 7.1 Attributes

### 7.1.1 name

`sequence.name`

#### Description

The name of the sequence.

#### Type

String; read/write.

---

### 7.1.2 id

`sequence.id`

#### Description

This is the ordinal assigned to the sequence, upon creation. If this is the thirty-third sequence created within the project during a given Premiere Pro session, this value will be ‘33’.

#### Type

Integer, read-only.

---

### 7.1.3 sequenceID

`sequence.sequenceID`

#### Description

The unique identifier assigned to this sequence, at the time of its creation.

#### Type

String; read-only.

---

### 7.1.4 audioTracks

`sequence.audioTracks`

#### Description

An array of audio tracks, within the sequence.

#### Type

Array; read-only.

---

### 7.1.5 videoTracks

`sequence.videoTracks`

#### Description

An array of video tracks, within the sequence.

#### Type

Array; read-only.

---

### 7.1.6 frameSizeHorizontal

`sequence.frameSizeHorizontal`

#### Description

The horizontal width of frames, from the sequence.

#### Type

Integer; read-only.

---

### 7.1.7 frameSizeVertical

```
sequence.frameSizeVertical
```

**Description**

The vertical height of frames, from the sequence.

**Type**

Integer; read-only.

---

### 7.1.8 timebase

```
sequence.timebase
```

**Description**

The number of Ticks per frame, in the sequence.

**Type**

Integer; read-only.

---

### 7.1.9 zeroPoint

```
sequence.zeroPoint
```

**Description**

The starting time, in Ticks, of the sequence.

**Type**

Integer; read-only.

---

### 7.1.10 setZeroPoint

```
sequence.setZeroPoint(newZeroPoint)
```

**Description**

Set the starting time of the sequence.

**Parameters**

An integer, specifying the new zero point, in Ticks.

**Type**

Integer; read-only.

**Returns**

Returns **0** if successful.

---

### 7.1.11 end

`sequence.end`

#### Description

The time, in Ticks, of the end of the sequence.

#### Type

Integer; read-only.

---

### 7.1.12 markers

`sequence.markers`

#### Description

The markers associated with this sequence.

#### Type

Array; read-only.

---

### 7.1.13 projectItem

`sequence.projectItem`

#### Description

The projectItem associated with this sequence.

#### Type

**projectItem**; read-only.

---

## 7.2 Methods

### 7.2.1 getPlayerPosition()

`sequence.getPlayerPosition()`

#### Description

Retrieves the current player position, in Ticks.

#### Parameters

None

#### Returns

Returns a Time object, representing the current player position.

---



### 7.2.2 `setPlayerPosition(newTimeInTicks)`

```
sequence.setPlayerPosition()
```

#### **Description**

Specifies a new player position, in Ticks.

#### **Parameters**

An integer, **newTimeInTicks**.

#### **Returns**

Returns **0** if successful.

---

### 7.2.3 `getInPoint()`

```
sequence.getInPoint()
```

#### **Description**

Retrieves the current sequence in point, in seconds.

#### **Parameters**

None.

#### **Returns**

Returns a Real representing the in point, in seconds.

---

### 7.2.4 `getOutPoint()`

```
sequence.getOutPoint()
```

#### **Description**

Retrieves the current sequence out point, in seconds.

#### **Parameters**

None.

#### **Returns**

Returns a Real representing the out point, in seconds.

---

### 7.2.5 `getInPointAsTime()`

```
sequence.getInPointAsTime()
```

#### **Description**

Retrieves the current sequence in point.

#### **Parameters**

None.

#### **Returns**

Returns a `Time` representing the in point, in seconds.

---

### 7.2.6 `getOutPointAsTime()`

```
sequence.getOutPointAsTime()
```

#### **Description**

Retrieves the current sequence out point.

#### **Parameters**

None.

#### **Returns**

Returns a `Time` representing the out point, in seconds.

---

### 7.2.7 `setInPoint(newTimeInTicks)`

```
sequence.setInPoint()
```

#### **Description**

Specifies a new sequence in point.

#### **Parameters**

An integer, `newTimeInTicks`.

#### **Returns**

Returns `0` if successful.

---

### 7.2.8 setOutPoint(newTimeInTicks)

```
sequence.setOutPoint()
```

**Description**

Specifies a new sequence out point.

**Parameters**

An integer, **newTimeInTicks**.

**Returns**

Returns **0** if successful.

---

### 7.2.9 clone()

```
sequence.clone()
```

**Description**

Creates a clone of the given sequence.

**Parameters**

None.

**Returns**

Returns a **Sequence** if successful, **0** if not.

---

### 7.2.10 exportAsProject(outputPath)

```
sequence.exportAsProject(outputPath)
```

**Description**

Creates a new project containing only the given sequence, and its constituent media.

**Parameters**

String **outputPath** specifying the output path for the new project.

**Returns**

Returns **0** if successful.

---

### 7.2.11 exportAsFinalCutProXML(outputPath)

```
sequence.exportAsFinalCutProXML(outputPath)
```

**Description**

Creates a new FCP XML representation of the sequence, and its constituent media.

**Parameters**

String `outputPath` specifying the output path for the new FCP XML file.

**Returns**

Returns 0 if successful.

---

### 7.2.12 exportAsMediaDirect(outputPath, presetPath, workAreaType)

```
sequence.exportAsMediaDirect(outputPath, presetPath, workAreaType)
```

**Description**

Renders the sequence to the specified output path, using the specified output preset (.epr file), and honoring the specified work area type.

**Parameters**

<code>outputPath</code>	<b>String</b> , Name of property to be added.
<code>presetPath</code>	<b>String</b> , Label of property to be added.
<code>workAreaType</code>	<b>Must be one of the following:</b> <ul style="list-style-type: none"><li>• 0 ENCODE_ENTIRE</li><li>• 1 ENCODE_IN_TO_OUT</li><li>• 2 ENCODE_WORK_AREA</li></ul>

String `outputPath` specifying the output path, to which to render the media.

**Returns**

Returns 0 if successful.

---

### 7.2.13 getExportFileExtension()

```
sequence.getExportFileExtension(outputPresetPath)
```

**Description**

Retrieves the file extension associated with the current sequence.

**Parameters**

String `outputPresetPath` specifying the output preset to be used.

**Returns**

Returns a **String** containing the output file extension, or **0** if unsuccessful.

---

### 7.2.14 `getSettings()`

`sequence.getSettings()`

#### **Description**

Retrieves the settings of the current sequence.

#### **Parameters**

None.

#### **Returns**

Returns a sequence settings structure.

audioChannelCount	The number of audio channels in the sequence.
audioChannelType	<p><b>Audio channel type in use. One of the following:</b></p> <ul style="list-style-type: none"> <li>• 0 AUDIOCHANNELTYPE_Mono</li> <li>• 1 AUDIOCHANNELTYPE_Stereo</li> <li>• 2 AUDIOCHANNELTYPE_51</li> <li>• 3 AUDIOCHANNELTYPE_Multichannel</li> <li>• 4 AUDIOCHANNELTYPE_4Channel</li> <li>• 5 AUDIOCHANNELTYPE_8Channel</li> </ul>
audioDisplayFormat	<p><b>Audio timecode display format. One of the following:</b></p> <ul style="list-style-type: none"> <li>• 100 TIMEDISPLAY_24Timecode</li> <li>• 101 TIMEDISPLAY_25Timecode</li> <li>• 102 TIMEDISPLAY_2997DropTimecode</li> <li>• 103 TIMEDISPLAY_2997NonDropTimecode</li> <li>• 104 TIMEDISPLAY_30Timecode</li> <li>• 105 TIMEDISPLAY_50Timecode</li> <li>• 106 TIMEDISPLAY_5994DropTimecode</li> <li>• 107 TIMEDISPLAY_5994NonDropTimecode</li> <li>• 108 TIMEDISPLAY_60Timecode</li> <li>• 109 TIMEDISPLAY_Frames</li> <li>• 110 TIMEDISPLAY_23976Timecode</li> <li>• 111 TIMEDISPLAY_16mmFeetFrames</li> <li>• 112 TIMEDISPLAY_35mmFeetFrames</li> <li>• 113 TIMEDISPLAY_48Timecode</li> <li>• 200 TIMEDISPLAY_AudioSamplesTimecode</li> <li>• 201 TIMEDISPLAY_AudioMsTimecode</li> </ul>
audioSampleRate	The audio sample rate in the sequence, as an int.
compositeLinearColor	Whether sequence is composited in linear color. 1 if true.
editingMode	The GUID of the editing mode in use.
maximumBitDepth	Whether sequence is composited at maximum depth; 1 if true.
maximumRenderQuality	Whether sequence is rendered at maximum quality; 1 if true.
previewCodec	Four character code of preview codec in use.
previewFrameWidth	Width of preview frame.
previewFrameHeight	Height of preview frame.
videoDisplayFormat	<p><b>Video time display format. One of the following:</b></p> <ul style="list-style-type: none"> <li>• 100 TIMEDISPLAY_24Timecode</li> <li>• 101 TIMEDISPLAY_25Timecode</li> <li>• 102 TIMEDISPLAY_2997DropTimecode</li> <li>• 103 TIMEDISPLAY_2997NonDropTimecode</li> <li>• 104 TIMEDISPLAY_30Timecode</li> <li>• 105 TIMEDISPLAY_50Timecode</li> <li>• 106 TIMEDISPLAY_5994DropTimecode</li> <li>• 107 TIMEDISPLAY_5994NonDropTimecode</li> <li>• 108 TIMEDISPLAY_60Timecode</li> <li>• 109 TIMEDISPLAY_Frames</li> <li>• 110 TIMEDISPLAY_23976Timecode</li> <li>• 111 TIMEDISPLAY_16mmFeetFrames</li> <li>• 112 TIMEDISPLAY_35mmFeetFrames</li> <li>• 113 TIMEDISPLAY_48Timecode</li> </ul>
50	<p><b>Chapter 70 Sequence object</b></p> <ul style="list-style-type: none"> <li>• 108 TIMEDISPLAY_60Timecode</li> <li>• 109 TIMEDISPLAY_Frames</li> <li>• 110 TIMEDISPLAY_23976Timecode</li> <li>• 111 TIMEDISPLAY_16mmFeetFrames</li> <li>• 112 TIMEDISPLAY_35mmFeetFrames</li> <li>• 113 TIMEDISPLAY_48Timecode</li> </ul>

### 7.2.15 setSettings()

```
sequence.setSettings(sequenceSettings)
```

#### Description

Sets the settings of the current sequence. *[Editorial: I apologize for any perceived pedantry; sometimes, obvious documentation needs to be obvious. -bbb]*

#### Parameters

`sequenceSettings` is a sequence settings structure, obtained via `sequence.getSettings_`.

#### Returns

Returns 0 if successful.

---

### 7.2.16 createSubsequence(ignoreChannelMapping)

```
sequence.createSubsequence(ignoreChannelMapping)
```

#### Description

Creates a new sequence, which is a sub-sequence of the existing sequence.

#### Parameters

A `Boolean` indicating whether the new sequence should ignore the channel mapping present in the original sequence.

#### Returns

Returns 0 if successful.





---

## Track object

---

Track

### Description

The **Track** object represents a video or audio track, within a sequence.

---

## 8.1 Attributes

### 8.1.1 name

`track.name`

#### Description

The name of the track.

#### Type

String; read-only.

---

### 8.1.2 id

`track.id`

#### Description

This is the ordinal assigned to the track, upon creation.

#### Type

Integer, read-only.

---

### 8.1.3 mediaType

`track.mediaType`

#### Description

The type of media, contained in this track.

#### Type

String, read-only; valid values are `Audio` and `Video`.

---

### 8.1.4 clips

`track.clips`

#### Description

An array of `trackItem` objects, contained within the track, in temporal order.

#### Type

Array; read-only.

---

### 8.1.5 transitions

`track.transitions`

#### Description

An array of transitions objects, contained within the track, in temporal order.

#### Type

Array; read-only.

---

## 8.2 Methods

### 8.2.1 isMuted()

`track.isMuted()`

#### Description

Retrieves the current mute state, of the track.

#### Parameters

None.

#### Returns

Returns **true** if track is currently muted; **false** if not.

---

### 8.2.2 setMute()

```
track.setMute(isMuted)
```

**Description**

Sets the mute state, of the track.

**Parameters**

Integer; if **1**, mute the track. If `isMuted` is **0**, the track will be unmuted.

**Returns**

Returns 0 if successful.

---

### 8.2.3 insertClip()

```
track.insertClip(srcProjectItem, time)
```

**Description**

Adds a 'clip' (media segment from a `projectItem`) to the track, at the specified time. Media will be inserted, at that time.

**Parameters**

A `projectItem` from which to get media, and the time at which to add it, in Ticks.

**Returns**

Returns 0 if successful.

---

### 8.2.4 overwriteClip()

```
track.overwriteClip(srcProjectItem, time)
```

**Description**

Adds a 'clip' (media segment from a `projectItem`) to the track, at the specified time. This will overwrite any existing media, at that time.

**Parameters**

A `projectItem` from which to get media, and the time at which to add it, in Ticks.

**Returns**

Returns 0 if successful.

---



---

## Track Item object

---

Track Item

### Description

The **trackItem** object represents an item on a video or audio track, within a sequence.

## 9.1 Attributes

### 9.1.1 name

`trackItem.name`

#### Description

The name of the track item.

#### Type

String; read/write.

---

### 9.1.2 duration

`trackItem.duration`

#### Description

The duration of the trackItem.

#### Type

Time object, read-only.

---

### 9.1.3 start

`trackItem.start`

#### **Description**

The starting time of the trackItem. Note: This may differ, from the trackItem's in point.

#### **Type**

Time object, read/write.

---

### 9.1.4 end

`trackItem.end`

#### **Description**

The ending time of the trackItem. Note: This may differ, from the trackItem's out point.

#### **Type**

Time object, read/write.

---

### 9.1.5 inPoint

`trackItem.inPoint`

#### **Description**

The in point for media, in this trackItem.

#### **Type**

Time object, read/write.

---

### 9.1.6 outPoint

`trackItem.outPoint`

#### **Description**

The out point for media, in this trackItem.

#### **Type**

Time object, read/write.

---

### 9.1.7 type

`trackItem.type`

#### Description

The type of media provided by this `trackItem`.

#### Type

1 means video, 2 means audio.

---

### 9.1.8 mediaType

`trackItem.mediaType`

#### Description

The `mediaType` of media provided by this `trackItem`.

#### Type

This will either be “**audio**” or “**video**”.

---

### 9.1.9 projectItem

`trackItem.projectItem`

#### Description

The `projectItem` from which the media is being drawn.

#### Type

A **`projectItem`**.

---

### 9.1.10 components

`trackItem.components`

#### Description

The components associated with this `trackItem`. This can include intrinsic transformations, as well as video and audio effects.

#### Type

An Array of components; read-only.

---

## 9.2 Methods

### 9.2.1 isSelected

```
trackItem.isSelected()
```

#### Description

Retrieves the current selection state of the trackItem.

#### Parameters

None.

#### Returns

Returns `true` if trackItem is selected; `false` if not.

---

### 9.2.2 setSelected

```
trackItem.setSelected(selectionState, updateUI)
```

#### Description

Sets the selection state of the trackItem.

#### Parameters

If selectionState is `1`, the trackItem will be selected; if `0`, it will be deselected. If updateUI is `1`, the Premiere Pro UI will be updated after this function call is made.

#### Returns

Returns `0` if successful.

---

### 9.2.3 isReversed

```
trackItem.isReversed()
```

#### Description

Returns whether the trackItem is reversed.

#### Parameters

None.

#### Returns

Returns `1` if trackItem is reversed; `0` if not.

---



## 9.2.4 `getSpeed`

```
trackItem.getSpeed()
```

### **Description**

Returns the speed multiplier applied to the `trackItem`.

### **Parameters**

None.

### **Returns**

Returns the speed multiplier applied to the `trackItem`, as a `float`. No speed adjustment = 1.

---

## 9.2.5 `isAdjustmentLayer`

```
trackItem.isAdjustmentLayer()
```

### **Description**

Returns whether the `trackItem` is an adjustment layer.

### **Parameters**

None.

### **Returns**

Returns `true` if the `trackitem` is an adjustment layer; `false` if not.



---

## Component object

---

Component

### Description

The **component** object represents something which has been added or applied to a `trackItem`.

---

## 10.1 Attributes

### 10.1.1 name

`component.displayName`

#### Description

The name of the component, as it is displayed to the user. Localized.

#### Type

String; read-only.

---

### 10.1.2 name

`component.matchName`

#### Description

The name of the component, as it is loaded from disk; used to uniquely identify effect plug-ins.

#### Type

String; read-only.

---

### 10.1.3 properties

`component.properties`

#### **Description**

The properties of the component in question; typically, these are effect parameters.

#### **Type**

Array of components, read-only.

---

## Component Parameter object

---

Component Parameter

### Description

The **component parameter** object represents a parameter associated with a component, applied to a trackItem.

## 11.1 Attributes

### 11.1.1 name

`componentParam.displayName`

### Description

The name of the component parameter, as it is displayed to the user. Localized.

### Type

String; read-only.

## 11.2 Methods

### 11.2.1 areKeyframesSupported

`componentParam.areKeyframesSupported()`

### Description

Retrieves whether keyframes are supported, for this component parameter.

### Parameters

None.

### Returns

Returns `true` if `trackItem` is selected; `false` if not.

---

## 11.2.2 `isTimeVarying`

```
componentParam.isTimeVarying()
```

### Description

Retrieves whether the component parameter varies, over time.

### Parameters

None.

### Returns

Returns `true` if the parameter varies over time; `false` if not.

---

## 11.2.3 `setTimeVarying`

```
componentParam.setTimeVarying(boolVary)
```

### Description

Sets whether the component parameter varies, over time. Note: This can only be set on parameters which support keyframing.

### Parameters

If `boolVary` is **true**, component parameter will vary over time; if **false**, it won't.

### Returns

Returns **0** if successful.

---

## 11.2.4 `findNearestKey`

```
componentParam.findNearestKey(timeToCheck, thresholdInTicks)
```

### Description

Sets whether the component parameter varies, over time. Note: This can only be set on parameters which support keyframing.

### Parameters

Starts search from `timeToCheck`, for `thresholdInTicks` temporal distance, in either direction.

### Returns

Returns a **Time** value, indicating when the closest keyframe is.

---

### 11.2.5 findPreviousKey

```
componentParam.findPreviousKey (timeToCheck)
```

#### Description

Returns the keyframe temporally previous to the provided `timeToCheck`. Note: This can only be set on parameters which support keyframing.

#### Parameters

Starts search from `timeToCheck`.

#### Returns

Returns a **Time** value, indicating when the closest keyframe is, or **0** if there is no available previous keyframe.

---

### 11.2.6 findNextKey

```
componentParam.findNextKey (timeToCheck)
```

#### Description

Returns the keyframe temporally subsequent to the provided `timeToCheck`. Note: This can only be set on parameters which support keyframing.

#### Parameters

Starts search from `timeToCheck`.

#### Returns

Returns a **Time** value, indicating when the closest keyframe is, or **0** if there is no available subsequent keyframe.

---

### 11.2.7 getKeys

```
componentParam.getKeys ()
```

#### Description

Returns an array of all keyframes on the `timeToCheck` component parameter. Note: This can only be set on parameters which support keyframing.

#### Parameters

None.

#### Returns

Returns an **Array** of **Time** values, indicating at what time each keyframe occurs, or **0** if no keyframes are available.

---

### 11.2.8 addKey

```
componentParam.addKey(time)
```

#### Description

Adds a keyframe to the component parameter stream, at the specified time. Note: This can only be set on parameters which support keyframing.

#### Parameters

A **Time** value, indicating when the keyframe should be added.

#### Returns

Returns **0** if successful.

---

### 11.2.9 removeKey

```
componentParam.removeKey(time)
```

#### Description

Removes a keyframe on the component parameter stream, at the specified time. Note: This can only be set on parameters which support keyframing.

#### Parameters

A **Time** value, indicating when the keyframe should be removed.

#### Returns

Returns **0** if successful.

---

### 11.2.10 removeKeyRange

```
componentParam.removeKeyRange(startTime, endTime)
```

#### Description

Removes all keyframes from the component parameter stream, between the specified times. Note: This can only be set on parameters which support keyframing.

#### Parameters

**Time** values, indicating at what times (inclusive) to begin and end the removal of keyframes from the component parameter stream.

#### Returns

Returns **0** if successful.

---



### 11.2.11 getValue

```
componentParam.getValue()
```

#### Description

Obtains the value of the component parameter stream. Note: This can only work on parameters which are not time-variant.

#### Parameters

None.

#### Returns

Returns the value of the component parameter stream; the return varies with stream type.

---

### 11.2.12 setValue

```
componentParam.setValue(newValue, boolUpdateUI)
```

#### Description

Obtains the value of the component parameter stream. Note: This can only work on parameters which are not time-variant.

#### Parameters

The `newValue` must be of the appropriate type for the component parameter stream; passing **1** for `boolUpdateUI` will force Premiere Pro to update its UI, after updating the value of the stream.

#### Returns

Returns **0** if successful.

---

### 11.2.13 getColorValue

```
componentParam.getColorValue()
```

#### Description

Obtains the value of the component parameter stream. Note: This can only work on parameters which are not time-variant.

#### Parameters

The `newValue` must be of the appropriate type for the component parameter stream; passing **1** for `boolUpdateUI` will force Premiere Pro to update its UI, after updating the value of the stream.

#### Returns

Returns a **Color** containing the values found in the component parameter stream, or **0** if unsuccessful.

---

### 11.2.14 setColorValue

```
componentParam.setColorValue(intAlpha, intRed, intGreen, intBlue,  
boolUpdateUI)
```

#### Description

Sets the values within a component parameter stream, representing a Color.

#### Parameters

Integers representing the alpha, red, green and blue values to be used in the component parameter stream; `boolUpdateUI` will force Premiere Pro to update its UI, after updating the value of the stream.

#### Returns

Returns **0** if successful.

---

### 11.2.15 getValueAtKey

```
componentParam.getValueAtKey(time)
```

#### Description

Retrieves the value of the component parameter stream, at the specified keyframe time. Note: Can only be used with keyframeable parameter streams.

#### Parameters

A `Time` from which the keyframe value should be retrieved;

#### Returns

Returns the value of the component parameter stream at `time`, or **0** if unsuccessful.

---

### 11.2.16 setValueAtKey

```
componentParam.setValueAtKey(time, newValue, boolUpdateUI)
```

#### Description

Sets the value of the component parameter stream, at the specified keyframe time. Note: Can only be used with keyframeable parameter streams.

#### Parameters

A `Time` at which the keyframe value should be set, and a `newValue` representing the value to be stored at the keyframe time; `boolUpdateUI` will force Premiere Pro to update its UI, after updating the value of the stream..

#### Returns

Returns **0** if successful.

---

### 11.2.17 setInterpolationTypeAtKey

```
componentParam.setInterpolationTypeAtKey(time, interpretationType)
```

#### Description

Specifies the interpolation typ to be assigned to the keyframe, at the specified time. Note: Can only be used with keyframeable parameter streams.

#### Parameters

A `Time` at which the interpretation type should be set (and which must correspond to an extant keyframe), and an `interpretationType` being set.

Time	<b>Time</b> of keyframe to modify.
interpretationType	<p><b>Must be one of the following:</b></p> <ul style="list-style-type: none"> <li>• 0 kfInterpMode_Linear</li> <li>• 1 kfInterpMode_EaseIn_Obsolete</li> <li>• 2 kfInterpMode_EaseOut_Obsolete</li> <li>• 3 kfInterpMode_EaseInEaseOut_Obsolete</li> <li>• 4 kfInterpMode_Hold</li> <li>• 5 kfInterpMode_Bezier</li> <li>• 6 kfInterpMode_Time</li> <li>• 7 kfInterpMode_TimeTransitionStart</li> <li>• 8 kfInterpMode_TimeTransitionEnd</li> </ul>

#### Returns

Returns **0** if successful.



anywhere

### Description

The **anywhere** object represents any Adobe Anywhere or Team Projects servers available.

## 12.1 Attributes

None.

## 12.2 Methods

### 12.2.1 setAuthenticationToken

```
anywhere.setAuthenticationToken(token, emailAddress)
```

#### Description

Logs the specified email address into the server, using the provided token.

#### Parameters

Takes an authorization `token`, and the associated email address.

#### Returns

Returns **0** if successful.

---

### 12.2.2 `getAuthenticationToken`

```
anywhere.getAuthenticationToken()
```

#### **Description**

Retrieves an authentication token.

#### **Parameters**

None.

#### **Returns**

A **String** containing the login token, or **0** if unsuccessful.

---

### 12.2.3 `isProductionOpen`

```
anywhere.isProductionOpen()
```

#### **Description**

Retrieves whether an Anywhere or Team Projects production is currently open.

#### **Parameters**

None.

#### **Returns**

Returns `true` if a production is open; `false` if not.

---

### 12.2.4 `listProductions`

```
anywhere.listProductions()
```

#### **Description**

Retrieves production names, available to the current user, on the current server.

#### **Parameters**

#### **Returns**

Returns an Array of **Strings** containing the names of available productions, or 0 if unsuccessful.

---

### 12.2.5 `openProduction`

```
anywhere.openProduction(productionURL)
```

#### **Description**

Opens the production at the specified URL.

#### **Parameters**

A **String** containing the url of the production to open.

---

**Returns**

Returns **0** if successful.

---

**12.2.6 getCurrentEditingSessionURL**

```
anywhere.getCurrentEditingSessionURL()
```

**Description**

Retrieves the URL of the Production, currently being edited.

**Parameters**

None.

**Returns**

Returns a **String** containing the production's URL, or **0** if unsuccessful.

---

**12.2.7 GetCurrentEditingSessionSelectionURL**

```
anywhere.GetCurrentEditingSessionSelectionURL()
```

**Description**

Retrieves the URL of the currently selected single asset. Will fail if more or fewer than one item is selected.

**Parameters**

None.

**Returns**

Returns a **String** containing the asset's URL, or **0** if unsuccessful (including if more or fewer than one item is selected).

---

**12.2.8 GetCurrentEditingSessionActiveSequenceURL**

```
anywhere.GetCurrentEditingSessionActiveSequenceURL()
```

**Description**

Retrieves the URL of the currently active sequence, within a production.

**Parameters**

None.

**Returns**

Returns a **String** containing the asset's URL, or **0** if unsuccessful (including if there is no active sequence, or if no editing session is opened).

---





encoder

### Description

The **encoder** object represents Adobe Media Encoder, and is used for local rendering, outside of Premiere Pro.

## 13.1 Attributes

None.

## 13.2 Methods

### 13.2.1 launchEncoder

`encoder.launchEncoder()`

#### Description

Launches Adobe Media Encoder.

#### Parameters

None.

#### Returns

Returns **0** if successful.

---

### 13.2.2 startBatch

```
encoder.startBatch()
```

**Description**

Makes Adobe Media Encoder start rendering its render queue.

**Parameters**

None.

**Returns**

Returns **0** if successful.

---

### 13.2.3 encodeSequence

```
encoder.encodeSequence(sequenceToRender, fullOutputPath, presetPath, workArea,  
boolRemoveUponCompletion)
```

**Description**

Makes Adobe Media Encoder render the specified sequence, with the specified settings.

**Parameters**

<code>sequenceToRender</code>	The <b>sequence</b> to render.
<code>fullOutputPath</code>	<b>String</b> , path to output file.
<code>presetPath</code>	<b>String</b> , path to preset (.epr) file.
<code>workArea</code>	<b>Integer denoting work area to be used:</b> <ul style="list-style-type: none"><li>• 0 ENCODE_ENTIRE</li><li>• 1 ENCODE_IN_TO_OUT</li><li>• 2 ENCODE_WORK_AREA</li></ul>
<code>boolRemoveUponCompletion</code>	If <b>1</b> , job will be removed once complete.

**Returns**

Returns a job ID as a **String**, for the render job added to the AME queue, or **0** if unsuccessful.

---

### 13.2.4 encodeProjectItem

```
encoder.encodeProjectItem(projectItem, fullOutputPath, presetPath, workArea,  
boolRemoveUponCompletion)
```

**Description**

Makes Adobe Media Encoder render (optionally, a specified range from) the specified projectItem, with the specified settings.

**Parameters**

<code>projectItem</code>	The <b>projectItem</b> to render.
<code>fullOutputPath</code>	<b>String</b> , path to output file.
<code>presetPath</code>	<b>String</b> , path to preset (.epr) file.
<code>workArea</code>	<b>Integer denoting work area to be used:</b> <ul style="list-style-type: none"> <li>• 0 ENCODE_ENTIRE</li> <li>• 1 ENCODE_IN_TO_OUT</li> <li>• 2 ENCODE_WORK_AREA</li> </ul>
<code>boolRemoveUponCompletion</code>	If <b>1</b> , job will be removed once complete.
<code>inPoint (optional)</code>	A <b>Time</b> , for the in point of new file.
<code>outPoint (optional)</code>	A <b>Time</b> , for the out point of new file.

**Returns**

Returns a job ID as a **String**, for the render job added to the AME queue, or **0** if unsuccessful.

**13.2.5 EncodeFile**

```
encoder.EncodeFile(fileToRender, fullOutputPath, presetPath, workArea,
boolRemoveUponCompletion)
```

**Description**

Makes Adobe Media Encoder render (optionally, a specified range from) the specified file, with the specified settings.

**Parameters**

<code>fileToRender</code>	<b>String</b> of file path, to render.
<code>fullOutputPath</code>	<b>String</b> , path to output file.
<code>presetPath</code>	<b>String</b> , path to preset (.epr) file.
<code>workArea</code>	<b>Integer denoting work area to be used:</b> <ul style="list-style-type: none"> <li>• 0 ENCODE_ENTIRE</li> <li>• 1 ENCODE_IN_TO_OUT</li> <li>• 2 ENCODE_WORK_AREA</li> </ul>
<code>boolRemoveUponCompletion</code>	If <b>1</b> , job will be removed once complete.
<code>inPoint</code>	A <b>Time</b> , for the in point of new file.
<code>outPoint</code>	A <b>Time</b> , for the out point of new file.

**Returns**

Returns a job ID as a **String**, for the render job added to the AME queue, or **0** if unsuccessful.

**13.2.6 setSidecarXMPEnabled**

```
encoder.setSidecarXMPEnabled(enabledOrNot)
```

**Description**

Determines whether a sidecar file containing XMP metadata, will be output.

**Parameters**

Pass **1** to enable sidecar output, **0** to disable.

**Returns**

Returns **0** if successful.

---

### 13.2.7 setEmbeddedXMPEnabled

```
encoder.setEmbeddedXMPEnabled(enabledOrNot)
```

**Description**

Determines whether embedded XMP metadata, will be output.

**Parameters**

Pass **1** to enable sidecar output, **0** to disable.

**Returns**

Returns **0** if successful.

Note: Premiere Pro and Adobe Media Encoder will output sidecar XMP for some file formats, and embed XMP for most. The applications make this determination based on numerous factors, and there is no API control to “force” sidecar or embedded output, for formats which normally use “the other approach”.

Marker

### **Description**

Both `projectItems` and `sequences` have associated **marker** objects, which represent their associated markers.

## **14.1 Attributes**

### **14.1.1 name**

`marker.name`

#### **Description**

The name of the marker.

#### **Type**

String; read/write.

---

### **14.1.2 comments**

`marker.comments`

#### **Description**

The comments within the marker.

#### **Type**

String; read/write.

---

### 14.1.3 type

`marker.type`

#### Description

The type of marker; either “Comment”, “Chapter”, “Segmentation”, or “WebLink”. Note: Premiere Pro can import some marker types, which cannot be created from within Premiere Pro.

#### Type

String; read-only.

---

### 14.1.4 guid

`marker.guid`

#### Description

The unique identifier of the marker, created at time of instantiation.

#### Type

String; read-only.

---

### 14.1.5 start

`marker.start`

#### Description

A **Time** object containing the value of the beginning of the marker.

#### Type

Time; read/write.

---

### 14.1.6 end

`marker.end`

#### Description

A **Time** object containing the value of the ending of the marker.

#### Type

Time; read/write.

---

## 14.2 Methods

### 14.2.1 setTypeAsComment

```
marker.setTypeAsComment ()
```

**Description**

Sets the type of the marker to “Comment”.

**Parameters**

None.

**Returns**

Returns **0** if successful.

---

### 14.2.2 setTypeAsChapter

```
marker.setTypeAsChapter ()
```

**Description**

Sets the type of the marker to “Chapter”.

**Parameters**

None.

**Returns**

Returns **0** if successful.

---

### 14.2.3 setTypeAsSegmentation

```
marker.setTypeAsSegmentation ()
```

**Description**

Sets the type of the marker to “Segmentation”.

**Parameters**

None.

**Returns**

Returns **0** if successful.

---

### 14.2.4 setTypeAsWebLink

```
marker.setTypeAsWebLink()
```

#### Description

Sets the type of the marker to “WebLink”.

#### Parameters

None.

#### Returns

Returns **0** if successful.

---

### 14.2.5 getWebLinkURL

```
marker.getWebLinkURL()
```

#### Description

Retrieves the URL, from the marker’s URL field.

#### Parameters

None.

#### Returns

Returns a *String* containing the URL, or **0** if unsuccessful.

---

### 14.2.6 getWebLinkFrameTarget

```
marker.getWebLinkFrameTarget()
```

#### Description

Retrieves the frame target, from the marker’s FrameTarget field.

#### Parameters

None.

#### Returns

Returns a *String* containing the frame target, or **0** if unsuccessful.

---



## 14.2.7 getColorByIndex

`marker.getColorByIndex(markerIndex)`

### Description

Gets the marker color index. (added in CC 2019)

### Parameters

<code>markerIndex</code>	Index of the marker to be read.
--------------------------	---------------------------------

### Returns

Returns the color index as an `Integer`.

## 14.2.8 setColorByIndex

`marker.setColorByIndex(colorIndex, markerIndex)`

### Description

Sets the marker color by index. Color indexes listed below. (added in CC 2019)

- 0 = Green
- 1 = Red
- 2 = Purple
- 3 = Orange
- 4 = Yellow
- 5 = White
- 6 = Blue
- 7 = Cyan

### Parameters

<code>colorIndex</code>	Index of the color to apply to the marker.
<code>markerIndex</code>	Index of the marker to be set.

### Returns

Returns `undefined`.



Source

### Description

The Source object represents Premiere Pro's Source monitor.

## 15.1 Attributes

None.

## 15.2 Methods

### 15.2.1 closeClip

```
source.closeClip()
```

#### Description

Closes the front-most clip in the Source monitor.

#### Parameters

None.

#### Returns

Returns **0** if successful.

---

### 15.2.2 closeAllClips

```
source.closeAllClips()
```

#### Description

Closes all clips in the Source monitor.

#### Parameters

None.

#### Returns

Returns **0** if successful.

---

### 15.2.3 getPosition

```
source.getPosition()
```

#### Description

Retrieves the position of the Source monitor's current time indicator.

#### Parameters

None.

#### Returns

Returns a `Time` object containing the position of the Source monitor's current time indicator.

---

### 15.2.4 openFilePath

```
source.openFilePath(absolutePathToFile)
```

#### Description

Open a file in the Source monitor.

#### Parameters

The absolute path to the file to open.

#### Returns

Returns **0** if successful.

---

### 15.2.5 openProjectItem

```
source.openProjectItem(projectItem)
```

#### Description

Open a project item in the Source monitor.

#### Parameters

The `projectItem` to open.

#### Returns

Returns 0 if successful.

---

### 15.2.6 play

```
source.play(playbackSpeed)
```

#### Description

Begins playing back the Source monitor, at the specified playback speed.

#### Parameters

A `float` representing the playback speed.

#### Returns

Returns 0 if successful.