Premiere Pro Scripting Guide

Release 14.5

Jun 03, 2021

Introduction

1 Introduction **1**

2 Changelog **3**
2.1 Adobe Premiere Pro 14.0 . . . 3
2.2 Adobe Premiere Pro 13.x . . . 3

3 Overview **5**
3.1 Example code . . . 5
3.2 Development and debugging tools . . . 5

4 How to Execute ExtendScript in Premiere Pro **7**

5 Application object **9**
5.1 Attributes . . . 9
5.2 Methods . . . 15

6 Anywhere object **23**
6.1 Attributes . . . 23
6.2 Methods . . . 23

7 Encoder object **27**
7.1 Attributes . . . 27
7.2 Methods . . . 27

8 Marker object **31**
8.1 Attributes . . . 31
8.2 Methods . . . 33

9 Metadata object **37**
9.1 Attributes . . . 37
9.2 Methods . . . 37

10 Production object **41**
10.1 Attributes . . . 41
10.2 Methods . . . 42

11 Project object **45**
11.1 Attributes . . . 45

11.2 Methods . . . 47

12 ProjectManager object **61**
12.1 Attributes . . . 61

13 Properties object **67**
13.1 Attributes . . . 67
13.2 Methods . . . 67

14 SourceMonitor object **71**
14.1 Attributes . . . 71
14.2 Methods . . . 71

15 ProjectItem object **75**
15.1 Attributes . . . 75
15.2 Methods . . . 78

16 TrackItem object **93**
16.1 Attributes . . . 93
16.2 Methods . . . 97

17 Component object **101**
17.1 Attributes . . . 101

18 ComponentParam object **103**
18.1 Attributes . . . 103
18.2 Methods . . . 104

19 Sequence object **113**
19.1 Attributes . . . 113
19.2 Methods . . . 117

20 Track object **127**
20.1 Attributes . . . 127
20.2 Methods . . . 129

21 AudioChannelMapping object **131**
21.1 Attributes . . . 131
21.2 Methods . . . 132

22 Time object **133**
22.1 Attributes . . . 133
22.2 Methods . . . 134

23 Collection object **137**
23.1 Objects . . . 137
23.2 Attributes . . . 137
23.3 Methods . . . 138

24 ComponentCollection object **139**
24.1 Attributes . . . 139

25 MarkerCollection object **141**
25.1 Attributes . . . 141
25.2 Methods . . . 142

26 ProjectCollection object **145**
26.1 Attributes 145

27 ProjectItemCollection object **147**
27.1 Attributes 147

28 SequenceCollection object **149**
28.1 Attributes 149

29 TrackCollection object **151**
29.1 Attributes 151

30 TrackItemCollection object **153**
30.1 Attributes 153

CHAPTER 1

Introduction

This reference is 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.

CHAPTER 2

Changelog

What's new and changed for scripting?

2.1 Adobe Premiere Pro 14.0

- **Scripting access to Auto-Reframe:**
  - Added: *Sequence.autoReframeSequence()*

2.2 Adobe Premiere Pro 13.x

- **Scripting access to Marker colors:**
  - Added: *Marker.getColorByIndex()*
  - Added: *Marker.setColorByIndex()*

CHAPTER 3

Overview

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 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.

3.1 Example code

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

3.2 Development and debugging tools

ExtendScript Toolkit (ESTK) is longer updated by Adobe; the recommended debugging environment for ExtendScript is Microsoft Visual Studio Code, with Adobe's ExtendScript debugging extension:

https://marketplace.visualstudio.com/items?itemName=Adobe.extendscript-debug

CHAPTER 4

How to Execute ExtendScript in Premiere Pro

Executing scripts from within CEP panels is the recommended approach.

With additional configuration work, it's also 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 VSCode with Adobe's extension for development and debugging; for deployed workflows, stick with panels whenever possible.

CHAPTER 5

Application object

`app`

Description

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

5.1 Attributes

5.1.1 app.anywhere

`app.anywhere`

Description

An *Anywhere object*, providing access to available Anywhere servers. Only available when running in Anywhere configuration (discontinued).

Type

Anywhere object.

5.1.2 app.build

`app.build`

Description

The number of the build of Premiere Pro being run.

Type

String; read-only.

Example

Get a build version of current application *(Adobe Premiere Pro version 14.3.1 (Build 45))*

```
parseInt(app.build); // 45
```

5.1.3 app.encoder

`app.encoder`

Description

Provides access to Adobe Media Encoder (on the same system).

Type

Encoder object.

5.1.4 app.getAppPrefPath

`app.getAppPrefPath`

Description

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

Type

String; read-only.

Example

Get a path to a currently active preference file

```
app.getAppPrefPath; // /Users/USERNAME/Documents/Adobe/Premiere Pro/14.0/Profile-
↪USERNAME/
```

5.1.5 app.getAppSystemPrefPath

`app.getAppSystemPrefPath`

Description

Premiere Pro's active configuration files, not specific to a given user.

Type

String; read-only.

Example

Get a path to a currently active configuration folder

```
app.getAppSystemPrefPath; // /Library/Application Support/Adobe/Adobe Premiere Pro␣
↪2020/
```

5.1.6 app.getPProPrefPath

`app.getPProPrefPath`

Description

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

Type

String; read-only.

Example

Get a path to a currently active preference file

```
app.getPProPrefPath; // /Users/USERNAME/Documents/Adobe/Premiere Pro/14.0/Profile-
↪USERNAME/
```

5.1.7 app.getPProSystemPrefPath

`app.getPProSystemPrefPath`

Description

Premiere Pro's active configuration files, not specific to a given user.

Type

String; read-only.

Example

Get a path to a currently active configuration folder

```
app.getPProSystemPrefPath; // /Library/Application Support/Adobe/Adobe Premiere Pro␣
↪2020/
```

5.1.8 app.learnPanelContentDirPath

`app.learnPanelContentDirPath`

Description

Get the Learn panel's contents directory path.

Type

String; read-only.

Example

Get a path to a Learn panel's directory

```
app.learnPanelContentDirPath; // /Users/Shared/Adobe/Premiere Pro 2020/Learn Panel/
```

5.1.9 app.learnPanelExampleProjectDirPath

`app.learnPanelExampleProjectDirPath`

Description

Get the Learn panel's example projects directory path.

Type

String; read-only.

Example

Get a path to a Learn panel's example projects' directory

```
app.learnPanelExampleProjectDirPath; // /Users/Shared/Adobe/Premiere Pro/14.0/
↪Tutorial/Going Home project/
```

5.1.10 app.metadata

`app.metadata`

Description

Get applications Metadata object.

Type

Metadata object, read-only.

5.1.11 app.path

`app.path`

Description

Get a path to applications executable file.

Type

String; read-only.

Example

Get a path to applications executable file.

```
app.path; // /Applications/Adobe Premiere Pro 2020/Adobe Premiere Pro 2020.app/
```

5.1.12 app.production

`app.production`

Description

The currently active production.

Type

Production object if at least 1 production is open, `null` otherwise.

5.1.13 app.project

`app.project`

Description

The currently active project.

Type

Project object.

5.1.14 app.projectManager

`app.projectManager`

Description

Provides access to project management functions within Premiere Pro.

Type

ProjectManager object.

5.1.15 app.projects

`app.projects`

Description

An array referencing all open projects; *numProjects* contains size.

Type

ProjectCollection object, read-only.

5.1.16 app.properties

`app.properties`

Description

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

Type

Properties object, read-only;

5.1.17 app.sourceMonitor

`app.sourceMonitor`

Description

Provides access to *SourceMonitor object*.

Type

SourceMonitor object.

5.1.18 app.userGuid

`app.userGuid`

Description

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

Type

String; read-only.

5.1.19 app.version

`app.version`

Description

The version of Premiere Pro, providing the API.

Type

String; read-only.

Example

Get a version of a current application *(Adobe Premiere Pro version 14.3.1 (Build 45))*

```
app.version; // 14.3.1
```

5.2 Methods

5.2.1 app.enableQE()

`app.enableQE()`

Description

Enables Premiere Pro's QE DOM.

Parameters

None.

Returns

Returns true if QE DOM was enabled.

5.2.2 app.getEnableProxies()

`app.getEnableProxies()`

Description

Determines whether proxy usage is currently enabled.

Parameters

None.

Returns

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

5.2.3 app.getWorkspaces()

`app.getWorkspaces()`

Description

Obtains an array of available workspaces as Strings.

Parameters

None.

Returns

`Array` if successful, `null` if unsuccessful.

Example

Get a list of available workspaces.

```
app.getWorkspaces();
/* [
    "All Panels",
    "Assembly",
    "Audio",
    "Color",
    "Editing",
    "Effects",
    "Graphics",
    "Learning",
    "Libraries",
    "Metalogging",
    "Production"
]; */
```

5.2.4 app.isDocument()

`app.isDocument(path)`

Description

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

Parameters

Argument	Type	Description
`path`	`String`	A path to a file.

Returns

Returns **true** if file can be opened as a Premiere Pro *project*.

Example

Test for valid project files

```
app.isDocument('~/Desktop/myProject.prproj'); // true
app.isDocument('~/Desktop/textFile.txt');     // false
app.isDocument('~/Desktop/footageFile.mov');  // false
app.isDocument('~/Desktop/imageFile.mov');    // false
```

5.2.5 app.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**.

5.2.6 app.newProject()

`app.newProject(path)`

Description

Creates a new .prproj *Project object*, at the specified path.

Parameters

Argument	Type	Description
`path`	`String`	A full path to new project; a .prproj extension will be added, if necessary.

Returns

Returns **true** if successful.

5.2.7 app.openDocument()

`app.openDocument(path)`

Description

Opens the file at the specified path, as a Premiere Pro *Project object*.

Parameters

Argument	Type	Description
`path`	`String`	Full path to the document to be opened.
`suppressConversionDialog`	`Boolean`	Optional. Suppress project conversion dialog.
`bypassLocateFileDialog`	`Boolean`	Optional. Bypass the locate file dialog.
`bypassWarningDialog`	`Boolean`	Optional. Bypass warning dialog.
`doNotAddToMRUList`	`Boolean`	Optional. Skip adding this file to the Most Recently Used List.

Returns

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

5.2.8 app.openFCPXML()

`app.openFCPXML(path, projPath)`

Description

Opens an FCP XML file as a Premiere Pro *Project object* (specified in projPath).

Parameters

Argument	Type	Description
`path`	`String`	
`projPath`	`String`	

Returns

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

5.2.9 app.quit()

```
app.quit()
```

Description

Quits Premiere Pro; user will be prompted to save any changes to *Project object*.

Parameters

None.

Returns

Nothing.

5.2.10 app.setEnableProxies()

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

Description

Determines whether proxy usage is currently enabled.

Parameters

Argument	Type	Description
`enabled`	`Integer`	`1` turns proxies on, `0` turns them off.

Returns

Returns 1 if proxy enablement was changed.

5.2.11 app.setExtensionPersistent()

```
app.setExtensionPersistent(extensionID, persistent)
```

Description

Whether extension with the given extensionID persists, within this session.

Parameters

Argument	Type	Description
`extensionID`	`String`	Which extension to modify.
`persistent`	`Integer`	Pass `1` to keep extension in memory, `0` to allow unloading.

Returns

Returns **true** if successful.

Example

```
var extensionID = 'com.adobe.PProPanel';
// 0 - while testing (to enable rapid reload);
// 1 - for "Never unload me, even when not visible."
var persistent = 0;

app.setExtensionPersistent(extensionID, persistent);
```

5.2.12 app.setScratchDiskPath()

`app.setScratchDiskPath(path, scratchDiskType)`

Description

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

Parameters

Argument	Type	Description
`path`	`String`	The new path to be used.
`scratchDiskType`	`Enum`	Enumerated value, must be one of the following: • `ScratchDiskType.FirstVideoCaptureFolder` • `ScratchDiskType.FirstAudioCaptureFolder` • `ScratchDiskType.FirstVideoPreviewFolder` • `ScratchDiskType.FirstAudioPreviewFolder` • `ScratchDiskType.FirstAutoSaveFolder` • `ScratchDiskType.FirstCCLibrariesFolder` • `ScratchDiskType.FirstCapsuleMediaFolder`

Returns

Returns 'true' if successful.

Example

```
var scratchPath = Folder.selectDialog('Choose new scratch disk folder');
if (scratchPath && scratchPath.exists) {
    app.setScratchDiskPath(scratchPath.fsName, ScratchDiskType.FirstAutoSaveFolder);
}
```

5.2.13 app.setSDKEventMessage()

`app.setSDKEventMessage(message, decorator)`

Description

Writes a string to Premiere Pro's Events panel.

Parameters

Argument	Type	Description
`message`	`String`	A message to display.

`decorator`	`String`	Decorator, one of:
		
		`info`
		`warning`
		`error`

Returns

Returns 'true' if successful.

5.2.14 app.setWorkspace()

`app.setWorkspace(workspace)`

Description

Set workspace as active. Use *app.getWorkspaces()* to get a list of all available workspaces.

Parameters

Argument	Type	Description
`workspace`	`String`	The name of the workspace.

Returns

`Boolean`.

Example

Activate *Editing* workspace.

```
var workspace = 'Editing';
if (app.setWorkspace(workspace)) {
    alert('Workspace changed to "' + workspace + '"');
} else {
    alert('Could not set "' + workspace + '" workspace');
}
```

5.2.15 app.trace()

`app.trace()`

Description

Writes a string to Premiere Pro's debug console.

Parameters

None.

Returns

Returns **true** if trace was added.

5.2.16 app.getProjectViewIDs()

`app.getProjectViewIDs()`

Description

Returns the view IDs of currently-open views, associated with any project.

Parameters

None.

Returns

An array of view IDs; can be null.

Example

```
var allViewIDs = app.getProjectViewIDs();
if (allViewIDs){
    var firstOne = allViewIDs[0];
} else {
    // No views open.
}
```

5.2.17 app.getProjectFromViewID()

`app.getProjectFromViewID()`

Description

Returns the Project associated with the provided View ID.

Parameters

A View ID, obtained from *getProjectViewIDs*.

Returns

A Project object, for the project associated with the provided View ID. Can be *null*.

Example

```
var allViewIDs = app.getProjectViewIDs();
if (allViewIDs){
    var firstOne = allViewIDs[0];
    if (firstOne){
        var thisProject = getProjectFromViewID(firstOne);
        if (thisProject){
            var name = thisProject.name;
        } else {
            // no project associated with that view ID.
        }
} else {
    // No views open.
}
```

CHAPTER 6

Anywhere object

`app.anywhere`

Description

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

6.1 Attributes

None.

6.2 Methods

6.2.1 Anywhere.getAuthenticationToken()

`app.anywhere.getAuthenticationToken()`

Description

Retrieves an authentication token.

Parameters

None.

Returns

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

6.2.2 Anywhere.getCurrentEditingSessionActiveSequenceURL()

```
app.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).

6.2.3 Anywhere.getCurrentEditingSessionSelectionURL()

```
app.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 fewre than one item is selected).

6.2.4 Anywhere.getCurrentEditingSessionURL()

```
app.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.

6.2.5 Anywhere.isProductionOpen()

`app.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.

6.2.6 Anywhere.listProductions()

`app.anywhere.listProductions()`

Description

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

Parameters

None.

Returns

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

6.2.7 Anywhere.openProduction()

`app.anywhere.openProduction(productionURL)`

Description

Opens the production at the specified URL.

Parameters

Argument	Type	Description
`productionURL`	`String`	The url of the production to open.

Returns

Returns **0** if successful.

6.2.8 Anywhere.setAuthenticationToken()

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

Description

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

Parameters

Argument	Type	Description
`token`	`String`	An authorization token.
`emailAddress`	`String`	The associated email address.

Returns

Returns **0** if successful.

CHAPTER 7

Encoder object

`app.encoder`

Description

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

7.1 Attributes

None.

7.2 Methods

7.2.1 Encoder.encodeFile()

```
app.encoder.encodeFile(filePath, outputPath, presetPath, workArea,
removeUponCompletion)
```

Description

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

Parameters

Argument	Type	Description
`filePath`	`String`	A path to a file to render.
`outputPath`	`String`	A path to an output file.
`presetPath`	`String`	A path to a preset (.epr) file.
`workArea`	`Integer`	Integer denoting work area to be used: • 0 - `ENCODE_ENTIRE` • 1 - `ENCODE_IN_TO_OUT` • 2 - `ENCODE_WORK_AREA`
`removeUponCompletion`	`Integer`	If `1`, job will be removed once complete.
`inPoint`	Time	A **Time**, for the in point of new file.
`outPoint`	Time	A **Time**, 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.

7.2.2 Encoder.encodeProjectItem()

```
app.encoder.encodeProjectItem(projectItem, outputPath, presetPath, workArea,
removeUponCompletion)
```

Description

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

Parameters

Argument	Type	Description
`projectItem`	*ProjectItem object*	A project item to render.
`outputPath`	`String`	A path to an output file.
`presetPath`	`String`	A path to a preset (.epr) file.
`workArea`	`Integer`	Integer denoting work area to be used: • 0 - `ENCODE_ENTIRE` • 1 - `ENCODE_IN_TO_OUT` • 2 - `ENCODE_WORK_AREA`
`removeUponCompletion`	`Integer`	If `1`, 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.

7.2.3 Encoder.encodeSequence()

`app.encoder.encodeSequence(sequence, outputPath, presetPath, workArea, removeUponCompletion)`

Description

Makes Adobe Media Encoder render the specified *Sequence object*, with the specified settings.

Parameters

Argument	Type	Description
`sequence`	*Sequence object*	A sequence to render.
`outputPath`	`String`	A path to an output file.
`presetPath`	`String`	A path to a preset (.epr) file.
`workArea`	`Integer`	Integer denoting work area to be used: • 0 - `ENCODE_ENTIRE` • 1 - `ENCODE_IN_TO_OUT` • 2 - `ENCODE_WORK_AREA`
`removeUponCompletion`	`Integer`	If `1`, 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.

7.2.4 Encoder.launchEncoder()

`app.encoder.launchEncoder()`

Description

Launches Adobe Media Encoder.

Parameters

None.

Returns

Returns **0** if successful.

7.2.5 Encoder.setEmbeddedXMPEnabled()

`app.encoder.setEmbeddedXMPEnabled(enabled)`

Description

Determines whether embedded XMP metadata, will be output.

Parameters

Argument	Type	Description
`enabled`	`Integer`	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".

7.2.6 Encoder.setSidecarXMPEnabled()

```
app.encoder.setSidecarXMPEnabled(enabled)
```

Description

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

Parameters

Argument	Type	Description
`enabled`	`Integer`	Pass `1` to enable sidecar output, `0` to disable.

Returns

Returns **0** if successful.

7.2.7 Encoder.startBatch()

```
app.encoder.startBatch()
```

Description

Makes Adobe Media Encoder start rendering its render queue.

Parameters

None.

Returns

Returns **0** if successful.

CHAPTER 8

Marker object

```
app.project.activeSequence.markers.getFirstMarker()
app.project.rootItem.children[index].getMarkers().getFirstMarker()
```

Description

Both *Project items* and *sequences* have associated **marker** objects, which represent their associated markers.

8.1 Attributes

8.1.1 Marker.comments

```
app.project.activeSequence.markers.getFirstMarker().comments
app.project.rootItem.children[index].getMarkers().getFirstMarker().comments
```

Description

The comments within the marker.

Type

String; read/write.

8.1.2 Marker.end

```
app.project.activeSequence.markers.getFirstMarker().end
app.project.rootItem.children[index].getMarkers().getFirstMarker().end
```

Description

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

Type

Time object; read/write.

8.1.3 Marker.guid

```
app.project.activeSequence.markers.getFirstMarker().guid
app.project.rootItem.children[index].getMarkers().getFirstMarker().guid
```

Description

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

Type

String; read-only.

8.1.4 Marker.name

```
app.project.activeSequence.markers.getFirstMarker().name
app.project.rootItem.children[index].getMarkers().getFirstMarker().name
```

Description

The name of the marker.

Type

String; read/write.

8.1.5 Marker.start

```
app.project.activeSequence.markers.getFirstMarker().start
app.project.rootItem.children[index].getMarkers().getFirstMarker().start
```

Description

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

Type

Time object; read/write.

8.1.6 Marker.type

```
app.project.activeSequence.markers.getFirstMarker().type
app.project.rootItem.children[index].getMarkers().getFirstMarker().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.

8.2 Methods

8.2.1 Marker.getColorByIndex()

```
app.project.activeSequence.markers.getFirstMarker().getColorByIndex(index)
app.project.rootItem.children[index].getMarkers().getFirstMarker().
getColorByIndex(index)
```

Note: This functionality was added in Adobe Premire Pro 13.x.

Description

Gets the marker color index.

Parameters

Argument	Type	Description
`index`	`Integer`	Index of the marker to be read.

Returns

Returns the color index as an `Integer`.

8.2.2 Marker.getWebLinkFrameTarget()

```
app.project.activeSequence.markers.getFirstMarker().getWebLinkFrameTarget()
app.project.rootItem.children[index].getMarkers().getFirstMarker().
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.

8.2.3 Marker.getWebLinkURL()

```
app.project.activeSequence.markers.getFirstMarker().getWebLinkURL()
app.project.rootItem.children[index].getMarkers().getFirstMarker().
getWebLinkURL()
```

Description

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

Parameters

None.

Returns

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

8.2.4 Marker.setColorByIndex()

```
app.project.activeSequence.markers.getFirstMarker().
setColorByIndex(colorIndex,
markerIndex)
app.project.rootItem.children[index].getMarkers().getFirstMarker().
setColorByIndex(colorIndex,
markerIndex)
```

Note: This functionality was added in Adobe Premire Pro 13.x.

Description

Sets the marker color by index. Color indexies listed below.

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

Parameters

Argument	Type	Description
`colorIndex`	`Integer`	Index of the color to apply to the marker.
`markerIndex`	`Integer`	Index of the marker to be set.

Returns

Returns `undefined`.

8.2.5 Marker.setTypeAsChapter()

```
app.project.activeSequence.markers.getFirstMarker().setTypeAsChapter()
app.project.rootItem.children[index].getMarkers().getFirstMarker().
setTypeAsChapter()
```

Description

Sets the type of the marker to "Chapter".

Parameters

None.

Returns

Returns **0** if successful.

8.2.6 Marker.setTypeAsComment()

```
app.project.activeSequence.markers.getFirstMarker().setTypeAsComment()
app.project.rootItem.children[index].getMarkers().getFirstMarker().
setTypeAsComment()
```

Description

Sets the type of the marker to "Comment".

Parameters

None.

Returns

Returns **0** if successful.

8.2.7 Marker.setTypeAsSegmentation()

```
app.project.activeSequence.markers.getFirstMarker().setTypeAsSegmentation()
app.project.rootItem.children[index].getMarkers().getFirstMarker().
setTypeAsSegmentation()
```

Description

Sets the type of the marker to "Segmentation".

Parameters

None.

Returns

Returns **0** if successful.

8.2.8 Marker.setTypeAsWebLink()

```
app.project.activeSequence.markers.getFirstMarker().setTypeAsWebLink()
app.project.rootItem.children[index].getMarkers().getFirstMarker().
setTypeAsWebLink()
```

Description

Sets the type of the marker to "WebLink".

Parameters

None.

Returns

Returns **0** if successful.

CHAPTER 9

Metadata object

`app.metadata`

Description

add description here

9.1 Attributes

9.1.1 Metadata.getMetadata

`app.metadata.getMetadata`

Description

add description here

Type

String.

9.2 Methods

9.2.1 Metadata.addMarker()

`app.metadata.addMarker()`

Description

add description here

Parameters

add parameters here

Returns

add return value/type here

9.2.2 Metadata.deleteMarker()

`app.metadata.deleteMarker()`

Description

add description here

Parameters

add parameters here

Returns

add return value/type here

9.2.3 Metadata.setMarkerData()

`app.metadata.setMarkerData()`

Description

add description here

Parameters

add parameters here

Returns

add return value/type here

9.2.4 Metadata.setMetadataValue()

`app.metadata.setMetadataValue()`

Description

add description here

Parameters

add parameters here

Returns

add return value/type here

9.2.5 Metadata.updateMarker()

```
app.metadata.updateMarker()
```

Description

add description here

Parameters

add parameters here

Returns

add return value/type here

CHAPTER 10

Production object

`app.production`

Description

The Production object lets ExtendScript access and manipulate productions, insert projects, create new projects and bins, and move existing Production projects to Trash.

10.1 Attributes

10.1.1 Production.name

`app.production.name`

Description

The name of the production.

Type

String.

10.1.2 Production.path

`app.production.path`

Description

The path to the Production folder.

Type

String.

10.1.3 Production.projects

`app.production.projects`

Description

An array of the projects containined within the Production, which are currently open. Does not include non-open projects.

Type

ProjectCollection object, read-only.

10.2 Methods

10.2.1 Production.addProject()

`app.production.addProject(srcProjectPath, destProjectPath)`

Description

Copies a project from some other location, into the Production directory.

Parameters

Argument	Type	Description
`srcProjectPath`	`String`	A path to the source project.
`destProjectPath`	`String`	A destination path for added project.

Returns

Returns **true** if successful.

10.2.2 Production.close()

`app.production.close()`

Description

Closes the Production, and all open projects from within that Production.

Parameters

None.

Returns

Returns **true** if successful.

10.2.3 Production.getLocked()

```
app.production.getLocked()
```

Description

Returns the current lock state of the Production.

Parameters

None.

Returns

Returns **true** if the Production is locked, **false** if it is unlocked.

10.2.4 Production.moveToTrash()

```
app.production.moveToTrash(projectOrFolderPath, suppressUI, saveProject)
```

Description

Moves the specified path ("bin") or .prproj into the Production's Trash folder.

Parameters

Argument	Type	Description
`projectOrFolderPath`	`String`	A path to the source project.
`suppressUI`	`Boolean`	Whether to suppress any resultant dialogues.
`saveProject`	`Boolean`	Whether to save the project(s) first.

Returns

Returns **true** if successful.

10.2.5 Production.setLocked()

```
app.production.setLocked(locked)
```

Description

Sets the lock state of the Production

Parameters

Argument	Type	Description
`locked`	`Boolean`	Desired lock state.

Returns

Returns **true** if successful.

CHAPTER 11

Project object

`app.project`

Description

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

11.1 Attributes

11.1.1 Project.activeSequence

`app.project.activeSequence`

Description

The currently active *Sequence object*, within the project.

Type

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

11.1.2 Project.cloudProjectlocalID

`app.project.cloudProjectlocalID`

Description

The ID of cloud project.

Type

String; read/only.

11.1.3 Project.documentID

`app.project.documentID`

Description

A unique identifier for this project, in format of `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.

Type

String; read-only.

11.1.4 Project.isCloudProject

`app.project.isCloudProject`

Description

Check whether the project is cloud project.

Type

Boolean; read-only.

11.1.5 Project.name

`app.project.name`

Description

The name of the project.

Type

String; read-only.

11.1.6 Project.path

`app.project.path`

Description

The file path of the project.

Type

String; read-only.

Example

Get a path of a curently active project

```
app.project.path; // /Users/USERNAME/Desktop/Project.prproj
```

11.1.7 Project.rootItem

`app.project.rootItem`

Description

A *ProjectItem object* representing the "root" of the project.

Type

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

11.1.8 Project.sequences

`app.project.sequences`

Description

The sequences within the project.

Type

SequenceCollection object, read-only.

11.2 Methods

11.2.1 Project.addPropertyToProjectMetadataSchema()

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

Description

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

Parameters

Argument	Type	Description
`propertyName`	`String`	A name of property to be added.
`propertyLabel`	`String`	A label of property to be added.
`propertyType`		Must be one of the following: • 0 `Integer` • 1 `Real` • 2 `String` • 3 `Boolean`

Returns

Returns **true** if successful, **undefined** if unsuccessful.

11.2.2 Project.closeDocument()

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

Description

Closes this project.

Parameters

Argument	Type	Description
`saveFirst`	`Integer`	If `1`, the project will be saved before closing.
`promptIfDirty`	`Integer`	If `1`, the user will be asked whether they want to save changes first.

Returns

Returns **0** if successful.

11.2.3 Project.consolidateDuplicates()

```
app.project.consolidateDuplicates()
```

Description

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

Parameters

None.

Returns

Returns **0** if successful.

11.2.4 Project.createNewSequence()

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

Description

Creates a new *Sequence object* with the specified ID.

Parameters

Argument	Type	Description
`sequenceName`	`String`	A name of a sequence.
`sequenceID`	`String`	An uniquely identifying ID for a new sequence.

Returns

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

11.2.5 Project.createNewSequenceFromClips()

```
app.project.createNewSequenceFromClips(sequenceName, arrayOfProjectItems,
destinationBin);
```

Description

Creates a new *Sequence object* with the given name, in the specified destination bin, and sequentially inserts project items into it.

Parameters

Argument	Type	Description
`sequenceName`	`String`	Optional. A name for a new sequence.
`arrayOfProjectItems`	`Array` of *ProjectItem* objects	An array of project items to be inserted into sequence.
`destinationBin`	*ProjectItem object*	Optional. A bin to contain sequence.

Returns

Returns the newly-created *Sequence object* if successful; *0* if unsuccessful.

11.2.6 Project.deleteSequence()

```
app.project.deleteSequence(sequence)
```

Description

Deletes the specified *Sequence object* from the project.

Parameters

Argument	Type	Description
`sequence`	*Sequence object*	A sequence to delete.

Returns

Returns 0 if successful.

11.2.7 Project.exportAAF()

```
app.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 object*, using the specified settings.

Parameters

Argument	Type	Description
`sequence`	*Sequence object*	A sequence to export.
`filePath`	`String`	An output path for .aaf file.
`mixdownVideo`	`Integer`	If `1`, render video before export.
`explodeToMono`	`Integer`	If `1`, breaks out stereo tracks to mono.
`sampleRate`		The sample rate of output audio.
`bitsPerSample`		The bits per sample of audio output.
`embedAudio`	`Integer`	If `1`, audio is embedded, if `0`, external.
`audioFileFormat`	`Integer`	`0` is AIFF, `1` is WAV.
`trimSources`	`Integer`	If `1`, trim audio files before export.
`handleFrames`	`Integer`	The number of handle frames (from 0 to 1000).
`presetPath`	`String`	A path to export preset (.epr) file.
`renderAudioEffects`	`Integer`	If `1`, render audio effects before export.
`includeClipCopies`	`Integer`	If `1`, include each copy of a clip.
`preserveParentFolder`	`Integer`	If `1`, preserves the parent folder, in output.

Returns

Returns **0** if successful.

11.2.8 Project.exportFinalCutProXML()

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

Description

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

Parameters

Argument	Type	Description
`outputPath`	`String`	An output path for .xml file.
`suppressUI`	`Integer`	If `1`, no warnings or alerts will be shown, during the export.

Returns

Returns 0 if successful.

11.2.9 Project.exportOMF()

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

Description

Exports an OMF file of the specified *Sequence object*, using the specified settings.

Parameters

Argument	Type	Description
`sequence`	*Sequence object*	The sequence to be output.
`filePath`	`String`	An output path for .omf file.
`omfTitle`	`String`	The title of the OMF.
`sampleRate`		The sample rate of output audio.
`bitsPerSample`		The bits per sample of audio output.
`audioEncapsulated`	`Integer`	If `1`, audio is embedded, if `0`, external.
`audioFileFormat`	`Integer`	`0` is AIFF, `1` is WAV.
`trimAudioFiles`	`Integer`	`1` means yes, trim audio files.
`handleFrames`	`Integer`	Number of handle frames (from 0 to 1000).
`includePan`	`Integer`	`1` means include pan info; `0` means don't.

Returns

Returns **0** if successful.

11.2.10 Project.exportTimeline()

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

Description

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

Parameters

Argument	Type	Description
`exportControllerName`	`String`	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.

11.2.11 Project.getGraphicsWhiteLuminance()

```
app.project.getGraphicsWhiteLuminance();
```

Description

Retrieves the current graphics white luminance value, for this project.

Parameters

None.

Returns

Returns the currently selected graphics white value.

11.2.12 Project.getInsertionBin()

```
app.project.getInsertionBin()
```

Description

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

Parameters

None.

Returns

Returns a *ProjectItem object* if successful, **0** if not.

11.2.13 Project.getProjectPanelMetadata()

```
app.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.

11.2.14 Project.getSharedLocation()

```
app.project.getSharedLocation()
```

Description

Returns the path to the location to which shared files are to be copied.

Parameters

None.

Returns

Returns a **String** containing the path.

11.2.15 Project.getSupportedGraphicsWhiteLuminances()

```
app.project.getSupportedGraphicsWhiteLuminances();
```

Description

Retrieves the supported graphics white luminance values, for this project.

Parameters

None.

Returns

Returns an array of graphics white settings supported by the project; Currently it returns (100, 203, 300)

11.2.16 Project.importAEComps()

```
app.project.importAEComps(path, compNames, targetBin)
```

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

Argument	Type	Description
`path`	`String`	A path to the After Effects .aep project file.
`compNames`	`Array`	Names of compositions within the specified project, to be imported.
`targetBin`	*ProjectItem object*	Optional. The destination bin for this import.

Returns

Returns **0** if successful.

11.2.17 Project.importAllAEComps()

```
app.project.importAllAEComps(path, targetBin)
```

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

Argument	Type	Description
`path`	`String`	A path to After Effects .aep project file.
`targetBin`	*ProjectItem object*	Optional. The destination bin for this import.

Returns

Returns **0** if successful.

11.2.18 Project.importFiles()

```
app.project.importFiles(filePaths, suppressUI, targetBin,
importAsNumberedStills)
```

Description

Imports media from the specified file paths.

Parameters

Argument	Type	Description
`filePaths`	`Array`	An array of the file paths to be imported.
`suppressUI`	`Boolean`	Whether warning dialogs should be suppressed.
`targetBin`	*ProjectItem object*	The bin into which the files should be imported.
`importAsNumberedStills`	`Boolean`	Whether the file paths should be interpreted as a sequence of numbered stills.

Returns

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

11.2.19 Project.importSequences()

```
app.project.importSequences(path, sequenceIDs)
```

Description

Imports an array of *sequence* objects (with specified sequenceIDs), from the specified project, into the current project.

Parameters

Argument	Type	Description
`path`	`String`	A path to a project file.
`sequenceIDs`	`Array`	An array of sequence IDs to import.

Returns

Returns **0** if successful.

11.2.20 Project.isSharedLocationCopyEnabled()

```
app.project.isSharedLocationCopyEnabled()
```

Description

Determines whether copying to a shared location is enabled, for this project.

Parameters

None.

Returns

Returns **true** if copying is enabled; **false** if not.

11.2.21 Project.newBarsAndTone()

```
app.project.newBarsAndTone(width, height, timeBase, PARNum, PARDen,
audioSampleRate, name)
```

Description

Creates a new *Sequence object* with the given name, based on the specified preset (.sqpreset file).

Parameters

Argument	Type	Description
`width`	`Integer`	
`height`	`Integer`	
`timeBase`		A timebase for a new project item.
`PARNum`	`Integer`	Pixel aspect ration numerator.
`PARDen`	`Integer`	Pixel aspect ration denominator.
`audioSampleRate`		Audio sample rate.
`name`	`String`	Name for a new project item.

Returns

Returns a *ProjectItem object* for the new bars and tone, or **0** if unsuccessful.

11.2.22 Project.newSequence()

```
app.project.newSequence(name, pathToSequencePreset)
```

Description

Creates a new *Sequence object* with the given name, based on the specified preset (.sqpreset file).

Parameters

Argument	Type	Description
`name`	`String`	Name for a new sequence.
`pathToSequencePreset`	`String`	A path to a preset .sqpreset file.

Returns

Returns a *Sequence object*, or **0** if unsuccessful.

11.2.23 Project.openSequence()

```
app.project.openSequence(sequence.sequenceID)
```

Description

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

Parameters

Argument	Type	Description
`sequenceID`	*Sequence.sequenceID*	A valid sequence ID that should be opened.

Returns

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

11.2.24 Project.pauseGrowing()

```
app.project.pauseGrowing(pause)
```

Description

Pauses (and resumes) growing file capture.

Parameters

Argument	Type	Description
`pause`	`Integer`	If `1`, growing files are enabled.

Returns

Returns **0** if successful.

11.2.25 Project.save()

```
app.project.save()
```

Description

Saves the project, at its current path.

Parameters

None.

Returns

Returns **0** if successful.

11.2.26 Project.saveAs()

```
app.project.saveAs(path)
```

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

Argument	Type	Description
`path`	`String`	A path to a new file.

Returns

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

11.2.27 Project.setEnableTranscodeOnIngest()

```
app.project.setEnableTranscodeOnIngest(state);
```

Description

Controls the enablement of transcode-upon-ingest behavior, for the given project.

Parameters

Argument	Type	Description
`state`	`Boolean`	The desired state.

Returns

Returns **true** if successful.

11.2.28 Project.setGraphicsWhiteLuminance()

```
app.project.setGraphicsWhiteLuminance(value)
```

Description

Sets the current graphics white luminance value, for this project.

Parameters

Argument	Type	Description
`value`	`Integer`	The value to be used; must be a value provided by *Project.getSupportedGraphicsWhiteLuminances()*.

Returns

Returns true if successful.

11.2.29 Project.setProjectPanelMetadata()

`app.project.setProjectPanelMetadata(layout)`

Description

Returns the current layout of the Project panel.

Parameters

Argument	Type	Description
`layout`	`String`	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.

11.2.30 Project.setScratchDiskPath()

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

Description

Changes the specified scratch disk path to a new path.

Parameters

Argument	Type	Description
`newPath`	`String`	A new path.
`scratchDiskType`	`Enum`	Enumerated value, must be one of the following: • `ScratchDiskType.FirstVideoCaptureFolder` • `ScratchDiskType.FirstAudioCaptureFolder` • `ScratchDiskType.FirstVideoPreviewFolder` • `ScratchDiskType.FirstAudioPreviewFolder` • `ScratchDiskType.FirstAutoSaveFolder` • `ScratchDiskType.FirstCCLibrariesFolder` • `ScratchDiskType.FirstCapsuleMediaFolder`

- `ScratchDiskType.FirstAudioCaptureFolder`
- `ScratchDiskType.FirstVideoPreviewFolder`
- `ScratchDiskType.FirstAudioPreviewFolder`

- `ScratchDiskType.FirstAutoSaveFolder`
- `ScratchDiskType.FirstCCLibrariesFolder`
- `ScratchDiskType.FirstCapsuleMediaFolder`

Returns

Returns **0** if unsuccessful.

CHAPTER 12

ProjectManager object

`app.projectManager.options`

Description

The ProjectManager object exposes Premiere Pro's Project Manager, for project consolidation, transfer and transcoding.

12.1 Attributes

12.1.1 ProjectManager.affectedSequences

`app.projectManager.options.affectedSequences`

Description

An *Array* of *Sequence* objects, to be exported.

Type

Array; read/write.

12.1.2 ProjectManager.clipTranscoderOption

`app.projectManager.options.clipTranscoderOption`

Description

The specified setting for clip transcode. Value will be one of the following:

CLIP_TRANSCODE_MATCH_PRESET	Transcode using the specified preset.
CLIP_TRANSCODE_MATCH_CLIPS	Match the clips
CLIP_TRANSCODE_MATCH_SEQUENCE | Must be one of the following:	

Type

String; read/write.

12.1.3 ProjectManager.clipTransferOption

`app.projectManager.options.clipTransferOption`

Description

The specified setting for clip transfer. Value will be one of the following:

CLIP_TRANSFER_COPY	Copy entire source media.
CLIP_TRANSFER_TRANSCODE	Transcode to default output format.

12.1.4 ProjectManager.convertAECompsToClips

`app.projectManager.options.convertAECompsToClips`

Description

If *true*, render dynamically-linked After Effects compositions to new media (using specified output preset).

Type

Boolean; read/write.

12.1.5 ProjectManager.convertImageSequencesToClips

`app.projectManager.options.convertImageSequencesToClips`

Description

If *true*, transcode image sequences to new media (using specified output preset).

Type

Boolean; read/write.

12.1.6 ProjectManager.convertSyntheticsToClips

`app.projectManager.options.convertSyntheticsToClips`

Description

If *true*, transcode clips from synthetic importers to new media (using specified output preset).

Type

Boolean; read/write.

12.1.7 ProjectManager.copyToPreventAlphaLoss

`app.projectManager.options.copyToPreventAlphaLoss`

Description

If *true*, includes any available alpha information into transcoded media.

Type

Boolean; read/write.

12.1.8 ProjectManager.destinationPath

`app.projectManager.options.destinationPath`

Description

The path to which to export the project and media.

Type

String; read/write.

12.1.9 ProjectManager.encoderPresetFilePath

`app.projectManager.options.encoderPresetFilePath`

Description

The path to the output preset (.epr file) to be used.

Type

String; read-write.

12.1.10 ProjectManager.excludeUnused

`app.projectManager.options.excludeUnused`

Description

If non-zero, exclude unused project items from the exported project.

Type

Boolean; read/write.

12.1.11 ProjectManager.handleFrameCount

`app.projectManager.options.handleFrameCount`

Description

How many frames of 'handle' footage (before and after the in and out points) of media, to include.

Type

Integer; read/write.

12.1.12 ProjectManager.includeAllSequences

`app.projectManager.options.includeAllSequences`

Description

If *true*, export all *Sequences* in the exported project.

Type

Boolean; read/write.

12.1.13 ProjectManager.includeConformedAudio

`app.projectManager.options.includeConformedAudio`

Description

If *true*, include conformed audio files with exported project.

Type

Boolean; read/write.

12.1.14 ProjectManager.includePreviews

`app.projectManager.options.includePreviews`

Description

If *true*, include rendered preview files with exported project.

Type

Boolean; read/write.

12.1.15 ProjectManager.renameMedia

`app.projectManager.options.renameMedia`

Description

If *true*, perform renaming as part of the export process.

Type

Boolean; read/write.

CHAPTER 13

Properties object

`app.properties`

Description

add description here

13.1 Attributes

None.

13.2 Methods

13.2.1 Properties.clearProperty()

`app.properties.clearProperty()`

Description

add description here

Parameters

add parameters here

Returns

add return value/type here

13.2.2 Properties.doesPropertyExist()

`app.properties.doesPropertyExist(property)`

Description

Checks whether a given property exists in preferences.

Parameters

Argument	Type	Description
`property`	`String`	A property to check

Returns

`Boolean`.

Example

Check whether labels with indices 10 and 99 exist in preferences:

```
var property = 'BE.Prefs.LabelNames.10';
var exists = app.properties.doesPropertyExist(property);
alert('Property "' + property + '" exists: ' + exists.toString());

property = 'BE.Prefs.LabelNames.99';
exists = app.properties.doesPropertyExist(property);
alert('Property "' + property + '" exists: ' + exists.toString());
```

13.2.3 Properties.getProperty()

`app.properties.getProperty(property)`

Description

Returns a property value.

Parameters

Argument	Type	Description
`property`	`String`	A property to get a value for

Returns

`String`.

Example

Get label name at a given index:

```
var labelIndex = 0;
var property = 'BE.Prefs.LabelNames.' + labelIndex;

if (app.properties.doesPropertyExist(property)) {
    alert(app.properties.getProperty(property));
} else {
    alert('Property "' + property + '" does not exist');
}
```

13.2.4 Properties.isPropertyReadOnly()

`app.properties.isPropertyReadOnly(property)`

Description

Checks whether a given property can be overwritten by the user. Returns *false* if such property does not exist.

Parameters

Argument	Type	Description
`property`	`String`	A property to check.

Returns

`Boolean`.

13.2.5 Properties.setProperty()

`app.properties.setProperty(property, value, persistent, createIfNotExist)`

Description

Set property value.

Parameters

Argument	Type	Description
`property`	`String`	A property to create
`value`	`Any`	A value for a property
`persistent`	`Boolean`	Whether if should be persistent between sessions
`createIfNotExist`	`Boolean`	Should create, if such property does not exist

Returns

`null`.

Example

Change label name:

```
var labelIndex = 0;
var property = 'BE.Prefs.LabelNamesX.' + labelIndex;

var newValue = 'Changed via Script';
var persistent = true;
var createIfNotExist = true;

if (app.properties.doesPropertyExist(property)) {
    if (app.properties.isPropertyReadOnly(property)) {
        alert('Could not rename property "' + property + '" because it is read-only.
↪');
```

(continues on next page)

(continued from previous page)

```
    } else {
        var oldValue = app.properties.getProperty(property);
        app.properties.setProperty(property, newValue, persistent, createIfNotExist);
        alert('Value changed from "' + oldValue + '" to "' + newValue + '"');
    }
} else {
    app.properties.setProperty(property, newValue, persistent, createIfNotExist);
    alert('Created new property "' + property + '" with value "' + newValue + '"');
}
```

CHAPTER 14

SourceMonitor object

`app.sourceMonitor`

Description

The Source object represents Premiere Pro's Source monitor.

14.1 Attributes

None.

14.2 Methods

14.2.1 SourceMonitor.closeAllClips()

`app.sourceMonitor.closeAllClips()`

Description

Closes all clips in the Source monitor.

Parameters

None.

Returns

Returns **0** if successful.

14.2.2 SourceMonitor.closeClip()

```
app.sourceMonitor.closeClip()
```

Description

Closes the front-most clip in the Source monitor.

Parameters

None.

Returns

Returns **0** if successful.

14.2.3 SourceMonitor.getPosition()

```
app.sourceMonitor.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.

14.2.4 SourceMonitor.openFilePath()

```
app.sourceMonitor.openFilePath(path)
```

Description

Open a file in the Source monitor.

Parameters

Argument	Type	Description
`path`	`String`	A path to the file to open.

Returns

Returns `true` if successful.

14.2.5 SourceMonitor.openProjectItem()

```
app.sourceMonitor.openProjectItem(projectItem)
```

Description

Open a project item in the Source monitor.

Parameters

Argument	Type	Description
`projectItem`	*ProjectItem object*	A project item to open.

Returns

Returns 0 if successful.

14.2.6 SourceMonitor.play()

```
app.sourceMonitor.play(playbackSpeed)
```

Description

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

Parameters

Argument	Type	Description
`playbackSpeed`	`Float`	The playback speed.

Returns

Returns 0 if successful.

CHAPTER 15

ProjectItem object

`app.project.rootItem.children[index]`

Description

Each item in a project is a **projectItem**, including the project root.

15.1 Attributes

15.1.1 ProjectItem.children

`app.project.rootItem.children[index].children`

Description

An array of project items, contained within the specified project item.

Type

ProjectItemCollection object, read-only.

15.1.2 ProjectItem.getAudioChannelMapping

`app.project.rootItem.children[index].getAudioChannelMapping`

Description

The audio channel mapping currently applied to this **projectItem**.

Type

An audioChannelMapping object.

15.1.3 ProjectItem.getOverrideColorSpaceList

`app.project.rootItem.children[index].getOverrideColorSpaceList`

Description

Add a description

Returns an object, containing similar data

```
{
    value: [
        sRGB,
        BT.601 (NTSC),
        BT.601 (PAL),
        BT.709,
        BT.709 (Scene),
        BT.2020,
        BT.2020 (Scene),
        BT.2100 PQ,
        BT.2100 PQ (Scene),
        BT.2100 HLG,
        BT.2100 HLG (Scene),
        DCDM XYZ,
    ]
};
```

Type

Javascript Object.

15.1.4 ProjectItem.name

`app.project.rootItem.children[index].name`

Description

The name of the project item.

Type

String; read/write.

Example

Rename first project item.

```
var item = app.project.rootItem.children[0];
if (item) {
    item.name = item.name + ', updated by PProPanel.';
} else {
    alert('Could not rename project item');
}
```

15.1.5 ProjectItem.nodeId

`app.project.rootItem.children[index].nodeId`

Description

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

NOTE: Distinguish between references to the same source media.

Type

String; read-only.

15.1.6 ProjectItem.teamProjectsAssetId

`app.project.rootItem.children[index].teamProjectsAssetId`

Description

The Team Projects Asset ID of the project item.

Type

String; read-only.

15.1.7 ProjectItem.treePath

`app.project.rootItem.children[index].treePath`

Description

The current project location of the project item. Example:

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

Type

String; read-only.

15.1.8 ProjectItem.type

`app.project.rootItem.children[index].type`

Description

Will be **CLIP**, **BIN**, **ROOT**, or **FILE**.

Type

Enumerated value; read-only.

15.2 Methods

15.2.1 ProjectItem.attachProxy()

```
app.project.rootItem.children[index].attachProxy(mediaPath, isHiRes)
```

Description

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

Parameters

Argument	Type	Description
`mediaPath`	`String`	The path to the the newly-assigned media.
`isHiRes`	`Integer`	Whether the new media should be attached as the proxy `0`, or high resolution `1` media.

Returns

Returns **0** if successful.

15.2.2 ProjectItem.canChangeMediaPath()

```
app.project.rootItem.children[index].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.

15.2.3 ProjectItem.canProxy()

```
app.project.rootItem.children[index].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.

15.2.4 ProjectItem.changeMediaPath()

`app.project.rootItem.children[index].changeMediaPath(newPath)`

Description

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

Parameters

Argument	Type	Description
`newPath`	`String`	A new path to the media file.
`overrideChecks`	`Boolean`	Override any safety concerns.

Returns

Returns **0** if replacement was successful.

15.2.5 ProjectItem.clearOutPoint()

`app.project.rootItem.children[index].clearOutPoint()`

Description

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

Parameters

None

Returns

Returns `0` if successful.

15.2.6 ProjectItem.createBin()

`app.project.rootItem.children[index].createBin(name)`

Description

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

Parameters

Argument	Type	Description
`name`	`String`	A name of a new bin.

Returns

Returns a project item representing the new bin if successful, or **0** if unsuccessful.

15.2.7 ProjectItem.createSmartBin()

```
app.project.rootItem.children[index].createSmartBin(name, queryString)
```

Description

Creates a search bin; only works for bin project items.

Parameters

Argument	Type	Description
`name`	`String`	A name of a new bin.
`queryString`	`String`	Query string for search.

Returns

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

15.2.8 ProjectItem.createSubClip()

```
app.project.rootItem.children[index].createSubClip(name, startTime, endTime,
hasHardBoundaries, takeAudio, takeVideo)
```

Description

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

Parameters

Argument	Type	Description
`name`	`String`	A name of a new subclip.
`startTime`	`String`	Start time of subclip, in **Ticks**.
`endTime`	`String`	End time of subclip, in **Ticks**.
`hasHardBoundaries`	`Integer`	If `1`, the user cannot extend *in* and *out*.
`takeAudio`	`Integer`	If `1`, use video from source.
`takeVideo`	`Integer`	If `1`, use video from source.

Returns

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

15.2.9 ProjectItem.deleteBin()

```
app.project.rootItem.children[index].deleteBin()
```

Description

Deletes a bin, **AND ALL ITS CONTENTS**, from the project.

Parameters

None.

Returns

Returns **0** if deletion was successful.

15.2.10 ProjectItem.findItemsMatchingMediaPath()

`app.project.rootItem.children[index].findItemsMatchingMediaPath(pathToMatch, ignoreSubClips)`

Description

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

Parameters

Argument	Type	Description
`pathToMatch`	`String`	A path to match.
`ignoreSubClips`	`Integer`	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.

15.2.11 ProjectItem.getColorLabel()

`app.project.rootItem.children[index].getColorLabel()`

Description

Retrieves the project item's color label.

Parameters

None.

Returns

`Number`, one of

labelColor	• 0 = Violet • 1 = Iris • 2 = Caribbean • 3 = Lavender • 4 = Cerulean • 5 = Forest • 6 = Rose • 7 = Mango • 8 = Purple • 9 = Blue • 10 = Teal • 11 = Magenta • 12 = Tan • 13 = Green • 14 = Brown • 15= Yellow

15.2.12 ProjectItem.getFootageInterpretation()

`app.project.rootItem.children[index].getFootageInterpretation()`

Description

Returns a structure describing the current interpretation of the projectItem.

Parameters

None.

Returns

A footage interpretation structure, or `0` if unsuccessful.

`alphaUsage`	**Alpha, will be one of the following:** • 0 ALPHACHANNEL_NONE • 1 ALPHACHANNEL_STRAIGHT • 2 ALPHACHANNEL_PREMULTIPLIED • 3 ALPHACHANNEL_IGNORE
`fieldType`	**Field type, one of the following:** • -1 FIELDTYPE_DEFAULT • 0 FIELDTYPE_PROGRESSIVE • 1 ALPHACHANNEL_UPPERFIRST • 2 ALPHACHANNEL_LOWERFIRST
`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`	**The projection type in use, for VR footage. One of these:** • 0 VR_CONFORM_PROJECTION_NONE • 1 VR_CONFORM_PROJECTION_EQUIRECTANGULAR
`vrLayoutType`	**The layout of footage in use, for VR. One of these:** • 0 VR_LAYOUT_MONOSCOPIC • 1 VR_LAYOUT_STEREO_OVER_UNDER • 2 VR_LAYOUT_STEREO_SIDE_BY_SIDE
`vrHorizontalView`	The horizontal view in use, for VR footage.
`vrVerticalView`	The vertical view in use, for VR footage.

15.2.13 ProjectItem.getInPoint()

```
app.project.rootItem.children[index].getInPoint()
```

Description

Obtains the current project item in point.

Parameters

None.

Returns

A *Time object*, containing the in point.

15.2.14 ProjectItem.getMarkers()

```
app.project.rootItem.children[index].getMarkers()
```

Description

Retrieves the *MarkerCollection object* associated with this project item.

Parameters

None.

Returns

MarkerCollection object, read-only;

15.2.15 ProjectItem.getMediaPath()

```
app.project.rootItem.children[index].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). Also, for image sequences, only the path to the first image in the sequence will be returned.

Parameters

None.

Returns

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

15.2.16 ProjectItem.getOutPoint()

`app.project.rootItem.children[index].getOutPoint(mediaType)`

Description

Retrieves the current out point for specified media type.

Parameters

Argument	Type	Description
`mediaType`	`Integer`	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*.

15.2.17 ProjectItem.getProjectMetadata()

`app.project.rootItem.children[index].getProjectMetadata()`

Description

Retrieves metadata associated with the project item. Distinct from media XMP.

Parameters

None.

Returns

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

15.2.18 ProjectItem.getProxyPath()

`app.project.rootItem.children[index].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.

15.2.19 ProjectItem.getXMPMetadata()

```
app.project.rootItem.children[index].getXMPMetadata()
```

Description

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

Parameters

None.

Returns

A String containing all XMP metadata, serialized.

15.2.20 ProjectItem.hasProxy()

```
app.project.rootItem.children[index].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.

15.2.21 ProjectItem.isMergedClip()

```
app.project.rootItem.children[index].isMergedClip()
```

Description

Indicates whether the project item refers to a merged clip.

Parameters

None.

Returns

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

15.2.22 ProjectItem.isMulticamClip()

```
app.project.rootItem.children[index].isMulticamClip()
```

Description

Indicates whether the project item refers to a multicam clip.

Parameters

None.

Returns

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

15.2.23 ProjectItem.isOffline()

```
app.project.rootItem.children[index].isOffline()
```

Description

Returns a Boolean indicating whether the project item is offline.

Parameters

None.

Returns

Boolean, `true` if offline.

15.2.24 ProjectItem.isSequence()

```
app.project.rootItem.children[index].isSequence()
```

Description

Indicates whether the project item refers to a *Sequence object*.

Parameters

None.

Returns

Returns `true` if the project item is a *Sequence object*, or a multicam clip, or a merged clip. Returns `false` if it isn't any of those.

15.2.25 ProjectItem.moveBin()

```
app.project.rootItem.children[index].moveBin(newParentBinProjectItem)
```

Description

Moves the projectItem into a new parent bin.

Parameters

None.

Returns

Returns **0** if move was successful.

15.2.26 ProjectItem.refreshMedia()

```
app.project.rootItem.children[index].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.

15.2.27 ProjectItem.renameBin()

```
app.project.rootItem.children[index].renameBin(newName)
```

Description

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

Parameters

Argument	Type	Description
`newName`	`String`	A new bin name.

Returns

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

15.2.28 ProjectItem.select()

`app.project.rootItem.children[index].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.

15.2.29 ProjectItem.setColorLabel()

`app.project.rootItem.children[index].setColorLabel(labelColor)`

Description

Sets the project item's color label.

Parameters

Argument	Type	Description
`labelColor`	`Integer`	A label color; see *ProjectItem.getColorLabel()*.

Returns

0 if successful.

15.2.30 ProjectItem.setFootageInterpretation()

`app.project.rootItem.children[index].setFootageInterpretation(interpretation)`

Description

Returns a structure describing the current interpretation of the projectItem.

Parameters

Argument	Type	Description
`interpretation`		A footage interpretation structure.

Returns

`0` if successful.

15.2.31 ProjectItem.setInPoint()

```
app.project.rootItem.children[index].setInPoint(time, mediaType)
```

Description

Sets the in point to `timeInTicks`, for specified media types.

Parameters

Argument	Type	Description
`time`	`String`	A time in **Ticks**.
`mediaType`	`Integer`	Determining which media type to affect; pass `1` for video only, `2` for audio only, or `4` for all media types.

Returns

Returns `0` if successful.

15.2.32 ProjectItem.setOffline()

```
app.project.rootItem.children[index].setOffline()
```

Description

Makes the project item offline.

Parameters

None.

Returns

`true` if successful.

15.2.33 ProjectItem.setOutPoint()

```
app.project.rootItem.children[index].setOutPoint(time, mediaType)
```

Description

Sets the out point to `timeInTicks`, for specified media types.

Parameters

Argument	Type	Description
`time`	`String`	A time in **Ticks**.
`mediaType`	`Integer`	Determining which media type to affect; pass `1` for video only, `2` for audio only, or `4` for all media types.

Returns

Returns `0` if successful.

15.2.34 ProjectItem.setOverrideFrameRate()

```
app.project.rootItem.children[index].setOverrideFrameRate(newFrameRate)
```

Description

Sets the frame rate of the project item.

Parameters

Argument	Type	Description
`newFrameRate`	`Float`	The new frame rate.

Returns

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

15.2.35 ProjectItem.setOverridePixelAspectRatio()

```
app.project.rootItem.children[index].setOverridePixelAspectRatio(numerator,
denominator)
```

Description

Sets the pixel aspect ratio for the project item.

Parameters

Argument	Type	Description
`numerator`	`Integer`	A new numerator.
`denominator`	`Integer`	A new denominator.

Returns

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

15.2.36 ProjectItem.setProjectMetadata()

```
app.project.rootItem.children[index].setProjectMetadata(newMetadata,
updatedFields)
```

Description

Sets the private project metadata associated with the project item.

Parameters

Argument	Type	Description
`newMetadata`	`String`	A new, serialized private project metadata.
`updatedFields`	`Array`	An array containing the names of the fields to be updated.

Returns

Returns 0 if update was successful.

15.2.37 ProjectItem.setScaleToFrameSize()

`app.project.rootItem.children[index].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.

15.2.38 ProjectItem.setStartTime()

`app.project.rootItem.children[index].setStartTime(time)`

Description

Assigns a new start time to the project item

Parameters

Argument	Type	Description
`time`	`String`	A new starting time, represented in **Ticks**.

Returns

Returns `0` if successful.

15.2.39 ProjectItem.setXMPMetadata()

`app.project.rootItem.children[index].setXMPMetadata(newXMP)`

Description

Sets the XMP metadata associated with the project item.

Parameters

Argument	Type	Description
`newXMP`	`String`	A new, serialized XMP metadata.

Returns

Returns 0 if update was successful.

15.2.40 ProjectItem.startTime()

```
app.project.rootItem.children[index].startTime()
```

Description

Returns a *Time object*, representing start time.

Parameters

None.

Returns

Time object.

15.2.41 ProjectItem.videoComponents()

```
app.project.rootItem.children[index].videoComponents()
```

Description

Video components for the 'Master Clip' of this project item.

Type

ComponentCollection object, read-only.

CHAPTER 16

TrackItem object

```
app.project.sequences[index].audioTracks[index].clips[index]
app.project.sequences[index].videoTracks[index].clips[index]
```

Description

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

16.1 Attributes

16.1.1 TrackItem.components

```
app.project.sequences[index].audioTracks[index].clips[index].components
app.project.sequences[index].videoTracks[index].clips[index].components
```

Description

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

Type

ComponentCollection object, read-only;

16.1.2 TrackItem.duration

```
app.project.sequences[index].audioTracks[index].clips[index].duration
app.project.sequences[index].videoTracks[index].clips[index].duration
```

Description

The duration of the trackItem.

Type

Time object, read-only.

16.1.3 TrackItem.end

```
app.project.sequences[index].audioTracks[index].clips[index].end
app.project.sequences[index].videoTracks[index].clips[index].end
```

Description

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

Type

Time object, read/write.

16.1.4 TrackItem.inPoint

```
app.project.sequences[index].audioTracks[index].clips[index].inPoint
app.project.sequences[index].videoTracks[index].clips[index].inPoint
```

Description

The in point for media, in this trackItem.

Type

Time object, read/write.

16.1.5 TrackItem.matchName

```
app.project.sequences[index].audioTracks[index].clips[index].matchName
app.project.sequences[index].videoTracks[index].clips[index].matchName
```

Description

Add a description

Type

String; read-only.

16.1.6 TrackItem.mediaType

```
app.project.sequences[index].audioTracks[index].clips[index].mediaType
app.project.sequences[index].videoTracks[index].clips[index].mediaType
```

Description

The mediaType of media provided by this trackItem.

Type

String, either **Audio** or **Video**.

16.1.7 TrackItem.name

```
app.project.sequences[index].audioTracks[index].clips[index].name
app.project.sequences[index].videoTracks[index].clips[index].name
```

Description

The name of the track item.

Type

String; read/write.

16.1.8 TrackItem.nodeId

```
app.project.sequences[index].audioTracks[index].clips[index].nodeId
app.project.sequences[index].videoTracks[index].clips[index].nodeId
```

Description

Add a description

Type

String.

16.1.9 TrackItem.outPoint

```
app.project.sequences[index].audioTracks[index].clips[index].outPoint
app.project.sequences[index].videoTracks[index].clips[index].outPoint
```

Description

The out point for media, in this trackItem.

Type

Time object, read/write.

16.1.10 TrackItem.projectItem

```
app.project.sequences[index].audioTracks[index].clips[index].projectItem
app.project.sequences[index].videoTracks[index].clips[index].projectItem
```

Description

The *ProjectItem object* from which the media is being drawn.

Type

A *ProjectItem object*.

16.1.11 TrackItem.start

```
app.project.sequences[index].audioTracks[index].clips[index].start
app.project.sequences[index].videoTracks[index].clips[index].start
```

Description

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

Type

Time object, read/write.

16.1.12 TrackItem.type

```
app.project.sequences[index].audioTracks[index].clips[index].type
app.project.sequences[index].videoTracks[index].clips[index].type
```

Description

The type of media provided by this trackItem.

Type

Number, **1** means video, **2** means audio.

16.2 Methods

16.2.1 TrackItem.getSpeed()

```
app.project.sequences[index].audioTracks[index].clips[index].getSpeed()
app.project.sequences[index].videoTracks[index].clips[index].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`.

16.2.2 TrackItem.isAdjustmentLayer()

```
app.project.sequences[index].audioTracks[index].clips[index].
isAdjustmentLayer()
app.project.sequences[index].videoTracks[index].clips[index].
isAdjustmentLayer()
```

Description

Returns wheter the `trackItem` is an adjustment layer.

Parameters

None.

Returns

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

16.2.3 TrackItem.isReversed()

```
app.project.sequences[index].audioTracks[index].clips[index].isReversed()
app.project.sequences[index].videoTracks[index].clips[index].isReversed()
```

Description

Returns whether the trackItem is reversed.

Parameters

None.

Returns

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

16.2.4 TrackItem.isSelected()

```
app.project.sequences[index].audioTracks[index].clips[index].isSelected()
app.project.sequences[index].videoTracks[index].clips[index].isSelected()
```

Description

Retrieves the current selection state of the trackItem.

Parameters

None.

Returns

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

16.2.5 TrackItem.setSelected()

```
app.project.sequences[index].audioTracks[index].clips[index].
setSelected(state,
updateUI)
app.project.sequences[index].videoTracks[index].clips[index].
setSelected(state,
updateUI)
```

Description

Sets the selection state of the trackItem.

Parameters

Argument	Type	Description
`state`	`Integer`	If `1`, the track item will be selected; if `0`, it will be deselected.
`updateUI`	`Integer`	If `1`, the Premiere Pro UI will be updated after this function call is made.

Returns

Returns **0** if successful.

16.2.6 TrackItem.getMatchName()

```
app.project.sequences[index].audioTracks[index].clips[index].getMatchName()
app.project.sequences[index].videoTracks[index].clips[index].getMatchName()
```

Description

Retrieves the match name for the trackItem.

Parameters

None.

Returns

Returns the match name as a **String** if successful.

16.2.7 TrackItem.remove()

```
app.project.sequences[index].audioTracks[index].clips[index].remove(inRipple,
inAlignToVideo)
app.project.sequences[index].videoTracks[index].clips[index].remove(inRipple,
inAlignToVideo)
```

Description

Sets the selection state of the trackItem.

Parameters

Argument	Type	Description
`inRipple`	`Boolean`	If `1`, later track items will be moved earlier, to fill the gap; if `0`, later track items will remain in place.
`inAlignToVideo`	`Boolean`	If `1`, Premiere Pro will align moved track items to the start of the nearest video frame.

Returns

Returns **0** if successful.

CHAPTER 17

Component object

```
app.project.sequences[index].audioTracks[index].clips[index].components[index]
app.project.sequences[index].videoTracks[index].clips[index].components[index]
```

Description

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

17.1 Attributes

17.1.1 Component.displayName

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].displayName
app.project.sequences[index].videoTracks[index].clips[index].
components[index].displayName
```

Description

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

Type

String; read-only.

17.1.2 Component.matchName

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].matchName
app.project.sequences[index].videoTracks[index].clips[index].
components[index].matchName
```

Description

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

Type

String; read-only.

17.1.3 Component.properties

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties
```

Description

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

Type

Array of components, read-only; (ComponentParamCollection object).

CHAPTER 18

ComponentParam object

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index]
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index]
```

Description

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

18.1 Attributes

18.1.1 ComponentParam.displayName

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].displayName
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].displayName
```

Description

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

Type

String; read-only.

18.2 Methods

18.2.1 ComponentParam.addKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].addKey(time)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].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

Argument	Type	Description
`time`	*Time object*	When the keyframe should be added.

Returns

Returns **0** if successful.

18.2.2 ComponentParam.areKeyframesSupported()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].areKeyframesSupported()
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].areKeyframesSupported()
```

Description

Retrieves whether keyframes are supported, for this component parameter.

Parameters

None.

Returns

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

18.2.3 ComponentParam.findNearestKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].findNearestKey(timeToCheck,
threshold)
```

```
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].findNearestKey(timeToCheck,
threshold)
```

Description

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

Parameters

Argument	Type	Description
`timeToCheck`		Start search from a given time
`threshold`		A temporal distance, in either direction, in **ticks**.

Returns

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

18.2.4 ComponentParam.findNextKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].findNextKey(timeToCheck)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].findNextKey(timeToCheck)
```

Description

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

Parameters

Argument	Type	Description
`timeToCheck`		Start search from a given time.

Returns

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

18.2.5 ComponentParam.findPreviousKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].findPreviousKey(timeToCheck)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].findPreviousKey(timeToCheck)
```

Description

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

Parameters

Argument	Type	Description
`timeToCheck`		Start search from a given time.

Returns

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

18.2.6 ComponentParam.getColorValue()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].getColorValue()
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].getColorValue()
```

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

18.2.7 ComponentParam.getKeys()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].getKeys()
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].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.

18.2.8 ComponentParam.getValue()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].getValue()
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].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.

18.2.9 ComponentParam.getValueAtKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].getValueAtKey(time)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].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

Argument	Type	Description
`time`	*Time object*	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.

18.2.10 ComponentParam.getValueAtTime()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].getValueAtTime(time)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].getValueAtTime(time)
```

Description

Retrieves the value of the component parameter stream, at the specified time. If the value is between two keyframes then interpolation takes place.

Parameters

Argument	Type	Description
`time`	*Time object*	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.

18.2.11 ComponentParam.isTimeVarying()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].isTimeVarying()
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].isTimeVarying()
```

Description

Retrieves whether the component parameter varies, over time.

Parameters

None.

Returns

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

18.2.12 ComponentParam.removeKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].removeKey(time)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].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

Argument	Type	Description
`time`	*Time object*	A time value, indicating when the keyframe should be removed.

Returns

Returns **0** if successful.

18.2.13 ComponentParam.removeKeyRange()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].removeKeyRange(startTime,
endTime)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].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

Argument	Type	Description
`startTime`	*Time object*	At what times (inclusive) to begin the removal of keyframes.
`endTime`	*Time object*	at what times to end the removal of keyframes.

Returns

Returns **0** if successful.

18.2.14 ComponentParam.setColorValue()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].setColorValue(alpha, red, green, blue,
updateUI)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].setColorValue(alpha, red, green, blue,
updateUI)
```

Description

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

Parameters

Argument	Type	Description
`alpha`	`Integer`	Alpha value.
`red`	`Integer`	Red value.
`green`	`Integer`	Green value.
`blue`	`Integer`	Blue value.
`updateUI`	`Integer`	Force to update UI after updating the value of the stream.

Returns

Returns **0** if successful.

18.2.15 ComponentParam.setInterpolationTypeAtKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].setInterpolationTypeAtKey(time,
interpretationType)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].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

Argument	Type	Description
`time`	*Time object*	A time of keyframe to modify.
`interpretationType`	`type`	Must be one of the following: • 0 `kfInterpMode_Linear` • 1 `kfInterpMode_EaseIn_Obsolete` • 2 `kfInterpMode_EaseOut_Obsolete` • 3 `kfInterpMode_EaseInEaseOut_Obsole` • 4 `kfInterpMode_Hold` • 5 `kfInterpMode_Bezier` • 6 `kfInterpMode_Time` • 7 `kfInterpMode_TimeTransitionStart` • 8 `kfInterpMode_TimeTransitionEnd`

Returns

Returns **0** if successful.

18.2.16 ComponentParam.setTimeVarying()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].setTimeVarying(varying)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].setTimeVarying(varying)
```

Description

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

Parameters

Argument	Type	Description
`varying`	`Boolean`	If `true`, component parameter will vary over time; if `false`, it won't.

Returns

Returns **0** if successful.

18.2.17 ComponentParam.setValue()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].setValue(value,
updateUI)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].setValue(value,
updateUI)
```

Description

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

Parameters

Argument	Type	Description
`value`		Must be of the appropriate type for the component parameter stream.
`updateUI`	`Integer`	If `1`, will force Premiere Pro to update UI, after updating the value of the stream.

Returns

Returns **0** if successful.

18.2.18 ComponentParam.setValueAtKey()

```
app.project.sequences[index].audioTracks[index].clips[index].
components[index].properties[index].setValueAtKey(time, value,
updateUI)
app.project.sequences[index].videoTracks[index].clips[index].
components[index].properties[index].setValueAtKey(time, value,
updateUI)
```

Description

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

Parameters

Argument	Type	Description
`time`	*Time object*	A time at which the keyframe value should be set.
`value`		A value to be set.
`updateUI`	`Integer`	If `1`, will force Premiere Pro to update UI, after updating the value of the stream.

Returns

Returns **0** if successful.

CHAPTER 19

Sequence object

`app.project.sequences[index]`

Description

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

19.1 Attributes

19.1.1 Sequence.audioDisplayFormat

`app.project.sequences[index].audioDisplayFormat`

Description

Add a description

Type

Number.

19.1.2 Sequence.audioTracks

`app.project.sequences[index].audioTracks`

Description

An array of audio tracks, within the sequence.

Type

TrackCollection object, read-only;

19.1.3 Sequence.end

`app.project.sequences[index].end`

Description

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

Type

String; read-only.

19.1.4 Sequence.frameSizeHorizontal

`app.project.sequences[index].frameSizeHorizontal`

Description

The horizontal width of frames, from the sequence.

Type

Integer; read-only.

19.1.5 Sequence.frameSizeVertical

`app.project.sequences[index].frameSizeVertical`

Description

The vertical height of frames, from the sequence.

Type

Integer; read-only.

19.1.6 Sequence.id

`app.project.sequences[index].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.

19.1.7 Sequence.markers

`app.project.sequences[index].markers`

Description

The *Marker* objects associated with this sequence.

Type

MarkerCollection object, read-only;

19.1.8 Sequence.name

`app.project.sequences[index].name`

Description

The name of the sequence.

Type

String; read/write.

19.1.9 Sequence.projectItem

`app.project.sequences[index].projectItem`

Description

The *ProjectItem object* associated with this sequence.

Type

projectItem; read-only.

19.1.10 Sequence.sequenceID

`app.project.sequences[index].sequenceID`

Description

The unique identifier assigned to this sequence, at the time of its creation, in form of `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.

Type

String; read-only.

19.1.11 Sequence.timebase

`app.project.sequences[index].timebase`

Description

The number of Ticks per frame, in the sequence.

Type

String; read-only.

19.1.12 Sequence.videoDisplayFormat

`app.project.sequences[index].videoDisplayFormat`

Description

Add a description

Type

Number.

19.1.13 Sequence.videoTracks

`app.project.sequences[index].videoTracks`

Description

An array of video tracks, within the sequence.

Type

TrackCollection object, read-only;

19.1.14 Sequence.zeroPoint

`app.project.sequences[index].zeroPoint`

Description

The starting time, in Ticks, of the sequence.

Type

String; read-only.

19.2 Methods

19.2.1 Sequence.autoReframeSequence()

`app.project.sequences[index].autoReframeSequence(numerator, denominator, motionPreset, newName, useNestedSequences)`

Description

Generates a new, auto-reframed sequence.

Parameters

Argument	Type	Description
`numerator`	`Integer`	Numerator of desired frame aspect ratio.
`denominator`	`Integer`	Denominator of desired frame aspect ratio.
`motionPreset`	`String`	One of: • "slower" • "default" • "faster"
`newName`	`String`	A name for a newly created sequence.
`useNestedSequences`	`Boolean`	Whether to honor nested sequence.

Returns

Returns the new *Sequence object*, if successful; *0* if unsuccessful.

Example

```
var sequence = app.project.activeSequence;
if (sequence) {
    var numerator = 1;
    var denominator = 1;
    var motionPreset = 'default'; // 'default', 'faster', 'slower'
    var newName = sequence.name + ', auto-reframed.';
    var useNestedSequences  = false;

    var newSequence = sequence.autoReframeSequence(numerator, denominator,␣
↪motionPreset, newName, useNestedSequences);

    if (newSequence) {
        alert('Created reframed sequence: ' + newName + '.');
    } else {
        alert('Failed to create re-framed sequence: ' + newName + '.');
    }
} else {
    alert('No active sequence');
}
```

19.2.2 Sequence.clone()

`app.project.sequences[index].clone()`

Description

Creates a clone of the given sequence.

Parameters

None.

Returns

Returns a boolean indicating whether the cloning was successful.

19.2.3 Sequence.createSubsequence()

`app.project.sequences[index].createSubsequence(ignoreChannelMapping)`

Description

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

Parameters

Argument	Type	Description
`ignoreChannelMappi`	`Boolean`	Whether the new sequence should ignore the channel mapping present in the original sequence.

Returns

Returns 0 if successful.

19.2.4 Sequence.exportAsFinalCutProXML()

`app.project.sequences[index].exportAsFinalCutProXML(outputPath)`

Description

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

Parameters

Argument	Type	Description
`outputPath`	`String`	The output path for the new FCP XML file.

Returns

Returns 0 if successful.

19.2.5 Sequence.exportAsMediaDirect()

`app.project.sequences[index].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

Argument	Type	Description
`outputPath`	`String`	An output path, to which to render the media.
`presetPath`	`String`	???
`workAreaType`		Must be one of the following: • 0 `ENCODE_ENTIRE` • 1 `ENCODE_IN_TO_OUT` • 2 `ENCODE_WORK_AREA`

Returns

Returns 0 if successful.

19.2.6 Sequence.exportAsProject()

`app.project.sequences[index].exportAsProject(outputPath)`

Description

Creates a new *Project object* containing only the given sequence, and its constituent media.

Parameters

Argument	Type	Description
`outputPath`	`String`	The output path for the new project.

Returns

Returns 0 if successful.

19.2.7 Sequence.getExportFileExtension()

`app.project.sequences[index].getExportFileExtension(outputPresetPath)`

Description

Retrieves the file extension associated with the current sequence.

Parameters

Argument	Type	Description
`outputPresetPath`	`String`	The output preset to be used.

Returns

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

19.2.8 Sequence.getInPoint()

```
app.project.sequences[index].getInPoint()
```

Description

Retrieves the current sequence in point, in seconds.

Parameters

None.

Returns

Returns a Real representing the in point, in seconds.

19.2.9 Sequence.getInPointAsTime()

```
app.project.sequences[index].getInPointAsTime()
```

Description

Retrieves the current sequence in point.

Parameters

None.

Returns

Returns a *Time object* representing the in point, in seconds.

19.2.10 Sequence.getOutPoint()

```
app.project.sequences[index].getOutPoint()
```

Description

Retrieves the current sequence out point, in seconds.

Parameters

None.

Returns

Returns a Real representing the out point, in seconds.

19.2.11 Sequence.getOutPointAsTime()

```
app.project.sequences[index].getOutPointAsTime()
```

Description

Retrieves the current sequence out point.

Parameters

None.

Returns

Returns a *Time object* representing the out point, in seconds.

19.2.12 Sequence.getPlayerPosition()

```
app.project.sequences[index].getPlayerPosition()
```

Description

Retrieves the current player position, in Ticks.

Parameters

None.

Returns

Returns a *Time object*, representing the current player position.

19.2.13 Sequence.getSettings()

```
app.project.sequences[index].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	**Audio channel type in use. One of the following:** • 0 AUDIOCHANNELTYPE_Mono • 1 AUDIOCHANNELTYPE_Stereo • 2 AUDIOCHANNELTYPE_51 • 3 AUDIOCHANNELTYPE_Multichannel • 4 AUDIOCHANNELTYPE_4Channel • 5 AUDIOCHANNELTYPE_8Channel
audioDisplayFormat	**Audio timecode display format. One of the following:** • 100 TIMEDISPLAY_24Timecode • 101 TIMEDISPLAY_25Timecode • 102 TIMEDISPLAY_2997DropTimecode • 103 TIMEDISPLAY_2997NonDropTimecode • 104 TIMEDISPLAY_30Timecode • 105 TIMEDISPLAY_50Timecode • 106 TIMEDISPLAY_5994DropTimecode • 107 TIMEDISPLAY_5994NonDropTimecode • 108 TIMEDISPLAY_60Timecode • 109 TIMEDISPLAY_Frames • 110 TIMEDISPLAY_23976Timecode • 111 TIMEDISPLAY_16mmFeetFrames • 112 TIMEDISPLAY_35mmFeetFrames • 113 TIMEDISPLAY_48Timecode • 200 TIMEDISPLAY_AudioSamplesTimecode • 201 TIMEDISPLAY_AudioMsTimecode
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.
previewFileFormat	Path to the output preset (.epr file) being used for preview file rendering.
videoDisplayFormat	**Video time display format. One of the following:** • 100 TIMEDISPLAY_24Timecode • 101 TIMEDISPLAY_25Timecode • 102 TIMEDISPLAY_2997DropTimecode • 103 TIMEDISPLAY_2997NonDropTimecode • 104 TIMEDISPLAY_30Timecode • 105 TIMEDISPLAY_50Timecode • 106 TIMEDISPLAY_5994DropTimecode • 107 TIMEDISPLAY_5994NonDropTimecode • 108 TIMEDISPLAY_60Timecode • 109 TIMEDISPLAY_Frames

19.2.14 Sequence.isDoneAnalyzingForVideoEffects()

`app.project.sequences[index].isDoneAnalyzingForVideoEffects()`

Description

Returns whether or not the sequence is done analyzing for video effects.

Parameters

None.

Returns

Returns `true` if analysis is complete.

19.2.15 Sequence.performSceneEditDetectionOnSelection()

`app.project.sequences[index].performSceneEditDetectionOnSelection(actionDesired, applyCutsToLinkedAudio, sensitivity)`

Description

Performs cut detection on the sequence selection.

Parameters

Argument	Type	Description
`actionDesired`	`String`	One of: • "CreateMarkers" • "ApplyCuts"
`applyCutsToLinkedAudio`	`Boolean`	
`sensitivity`	`String`	One of: • "LowSensitivity" • "MediumSensitivity" • "HighSensitivity"

Returns

Returns *true* if successful.

19.2.16 Sequence.setInPoint()

`app.project.sequences[index].setInPoint(time)`

Description

Specifies a new sequence in point.

Parameters

Argument	Type	Description
`time`	`String`	A new time in **ticks**.

Returns

Returns **0** if successful.

19.2.17 Sequence.setOutPoint()

```
app.project.sequences[index].setOutPoint(time)
```

Description

Specifies a new sequence out point.

Parameters

Argument	Type	Description
`time`	`String`	A new time in **ticks**.

Returns

Returns **0** if successful.

19.2.18 Sequence.setPlayerPosition()

```
app.project.sequences[index].setPlayerPosition(time)
```

Description

Specifies a new player position, in Ticks, as a String.

Parameters

Argument	Type	Description
`time`	`String`	A new time in **ticks**.

Returns

Returns **0** if successful.

19.2.19 Sequence.setSettings()

`app.project.sequences[index].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

Argument	Type	Description
`sequenceSettings`		A sequence settings structure, obtained via *Sequence.getSettings()*.

Returns

Returns 0 if successful.

19.2.20 Sequence.setZeroPoint()

`app.project.sequences[index].setZeroPoint(newZeroPoint)`

Description

Set the starting time of the sequence.

Parameters

Argument	Type	Description
`newZeroPoint`	`String`	The new zero point in **ticks**.

Type

Integer; read-only.

Returns

Returns **0** if successful.

CHAPTER 20

Track object

```
app.project.sequences[index].audioTracks[index]
app.project.sequences[index].videoTracks[index]
```

Description

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

20.1 Attributes

20.1.1 Track.clips

```
app.project.sequences[index].audioTracks[index].clips
app.project.sequences[index].videoTracks[index].clips
```

Description

An array of *Track item* objects, contained within the track, in temporal order.

Type

TrackItemCollection object, read-only;

20.1.2 Track.id

```
app.project.sequences[index].audioTracks[index].id
app.project.sequences[index].videoTracks[index].id
```

Description

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

Type

Integer, read-only.

20.1.3 Track.mediaType

```
app.project.sequences[index].audioTracks[index].mediaType
app.project.sequences[index].videoTracks[index].mediaType
```

Description

The type of media, contained in this track.

Type

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

20.1.4 Track.name

```
app.project.sequences[index].audioTracks[index].name
app.project.sequences[index].videoTracks[index].name
```

Description

The name of the track.

Type

String; read-only.

20.1.5 Track.transitions

```
app.project.sequences[index].audioTracks[index].transitions
app.project.sequences[index].videoTracks[index].transitions
```

Description

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

Type

TrackItemCollection object, read-only;

20.2 Methods

20.2.1 Track.insertClip()

```
app.project.sequences[index].audioTracks[index].insertClip(projectItem, time)
app.project.sequences[index].videoTracks[index].insertClip(projectItem, time)
```

Description

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

Parameters

Argument	Type	Description
`projectItem`	*ProjectItem object*	A project item from which to get media.
`time`	`String`	The time at which to add project item, in **Ticks**.

Returns

None.

20.2.2 Track.isMuted()

```
app.project.sequences[index].audioTracks[index].isMuted()
app.project.sequences[index].videoTracks[index].isMuted()
```

Description

Retrieves the current mute state, of the track.

Parameters

None.

Returns

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

20.2.3 Track.overwriteClip()

```
app.project.sequences[index].audioTracks[index].overwriteClip(projectItem,
time)
app.project.sequences[index].videoTracks[index].overwriteClip(projectItem,
time)
```

Description

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

Parameters

Argument	Type	Description
`projectItem`	*ProjectItem object*	A project item from which to get media.
`time`	`String`	The time at which to add project item, in **Ticks**.

Returns

Returns `true`.

20.2.4 Track.setMute()

```
app.project.sequences[index].audioTracks[index].setMute(isMuted)
app.project.sequences[index].videoTracks[index].setMute(isMuted)
```

Description

Sets the mute state, of the track.

Parameters

Argument	Type	Description
`isMuted`	`Integer`	If `1`, mute the track. If `0`, the track will be unmuted.

Returns

Returns 0 if successful.

CHAPTER 21

AudioChannelMapping object

`app.project.rootItem.children[index].getAudioChannelMapping`

Description

The AudioChannelMapping object defines the audio channel mapping applied to a given *ProjectItem object*.

21.1 Attributes

21.1.1 AudioChannelMapping.audioChannelsType

`app.project.rootItem.children[index].getAudioChannelMapping.audioChannelsType`

Description

The type of the audio contained in this channel. Will be 0, 1 or 2, corresponding to `AUDIOCHANNELTYPE_Mono`, `AUDIOCHANNELTYPE_Stereo`, or `AUDIOCHANNELTYPE_51`.

21.1.2 AudioChannelMapping.audioClipsNumber

`app.project.rootItem.children[index].getAudioChannelMapping.audioClipsNumber`

Description

The number of audio clips associated with this audio channel.

21.2 Methods

21.2.1 AudioChannelMapping.setMappingForChannel()

`app.project.rootItem.children[index].setMappingForChannel(channelIndex, sourceChannelIndex)`

Description

Maps a source channel to the specified channel index.

Parameters

Argument	Type	Description
`channelIndex`	`Integer`	The index of a channel to be mapped.
`sourceChannelIndex`	`Integer`	The index of a source channel to map.

Returns

Returns **true** if successful, **false** if that mapping is unsupported.

CHAPTER 22

Time object

```
myTime = new Time();
```

Description

An object representing a time. Internally, the time is computed in `ticks`; there are 254016000000 ticks per second. That time can be accessed in different representations, including as a timecode string.

22.1 Attributes

22.1.1 Time.seconds

```
myTime.seconds
```

Description

The time value, expressed in seconds.

Type

Number.

22.1.2 Time.ticks

```
myTime.ticks
```

Description

The time value, expressed in ticks.

Type

String.

22.2 Methods

22.2.1 Time.getFormatted()

```
myTime.getFormatted(frameRate, displayFormat)
```

Description

Returns the value of the `Time` passed, as a string, formatted in the specified display format.

Parameters

Argument	Type	Description
`frameRate`	`String`	The frame rate to be used, for the String-based time value.
`displayFormat`	`int`	The display format to use. Will be one of the following: TIMEDISPLAY_24Timecode = 100; TIMEDISPLAY_25Timecode = 101; TIMEDISPLAY_2997DropTimecode = 102; TIMEDISPLAY_2997NonDropTimecode = 103; TIMEDISPLAY_30Timecode = 104; TIMEDISPLAY_50Timecode = 105; TIMEDISPLAY_5994DropTimecode = 106; TIMEDISPLAY_5994NonDropTimecode = 107; TIMEDISPLAY_60Timecode = 108; TIMEDISPLAY_Frames = 109; TIMEDISPLAY_23976Timecode = 110; TIMEDISPLAY_16mmFeetFrames = 111; TIMEDISPLAY_35mmFeetFrames = 112; TIMEDISPLAY_48Timecode = 113; TIMEDISPLAY_AudioSamplesTimecode = 200; TIMEDISPLAY_AudioMsTimecode = 201;

Returns

A `String`.

22.2.2 Time.setSecondsAsFraction()

```
myTime.setSecondsAsFraction(numerator, denominator)
```

Description

Sets the Time object to the result of dividing the numerator by the denominator.

Parameters

Both the numerator and the denominator are `int`s.

Returns

Boolean; `true` if successful.

CHAPTER 23

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.

23.1 Objects

- *ComponentCollection object* - *todo*.
- *MarkerCollection object* - a collection of the *Marker objects* in a *ProjectItem object* and *Sequence object*.
- *ProjectCollection object* - a collection of *Project objects*.
- *ProjectItemCollection object* - a collection of *ProjectItem objects*.
- *SequenceCollection object* - a collection of *Sequence objects*.
- *TrackCollection object* - a collection of *Track objects*.
- *TrackItemCollection object* - a collection of *TrackItem objects*.

23.2 Attributes

`length`	The number of objects in the collection.

23.3 Methods

[]	Retrieves an object in the collection by its index number. The first object is at index 0.

CHAPTER 24

ComponentCollection object

```
app.project.rootItem.children[index].videoComponents()
app.project.sequences[index].audioTracks[index].clips[index].components
app.project.sequences[index].videoTracks[index].clips[index].components
```

add a description

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

24.1 Attributes

24.1.1 ComponentCollection.numItems

```
app.project.rootItem.children[index].videoComponents().numItems
app.project.sequences[index].audioTracks[index].clips[index].components.
numItems
app.project.sequences[index].videoTracks[index].clips[index].components.
numItems
```

Description

add a description

Type

Integer, read-only.

CHAPTER 25

MarkerCollection object

```
app.project.sequences[index].markers
app.project.rootItem.children[index].getMarkers()
```

The MarkerCollection object represents a collection of *Marker objects* in a *ProjectItem object* and *Sequence object*.

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

25.1 Attributes

25.1.1 MarkerCollection.numMarkers

```
app.project.sequences[index].markers.numMarkers
app.project.rootItem.children[index].getMarkers().numMarkers
```

Description

The count of marker objects in the project item or sequence.

Type

Integer, read-only.

25.2 Methods

25.2.1 MarkerCollection.createMarker()

```
app.project.sequences[index].markers.createMarker(time)
app.project.rootItem.children[index].getMarkers().createMarker(time)
```

Description

Create a new *Marker object* on a project item or a sequence.

Parameters

Argument	Type	Description
`time`	`Float`	A time, in seconds, where marker should be created.

Returns

Marker object if successful.

25.2.2 MarkerCollection.deleteMarker()

```
app.project.sequences[index].markers.deleteMarker(marker)
app.project.rootItem.children[index].getMarkers().deleteMarker(marker)
```

Description

Remove a given marker object from a collection.

Parameters

Argument	Type	Description
`marker`	*Marker object*	A marker object to remove from collection.

Returns

`Boolean`.

Examples

Remove all markers from the active sequence

```
var markers = app.project.activeSequence.markers;
var marker = markers.getFirstMarker();
var count = markers.numMarkers;

while (marker) {
    markers.deleteMarker(marker);
    marker = markers.getFirstMarker();
}

alert('Removed ' + count.toString() + ' markers');
```

25.2.3 MarkerCollection.getFirstMarker()

```
app.project.sequences[index].markers.getFirstMarker()
app.project.rootItem.children[index].getMarkers().getFirstMarker()
```

Description

Retrieve the first marker object, sorted by time in seconds, on a given project item or sequence.

Parameters

None.

Returns

Marker object or `undefined`.

25.2.4 MarkerCollection.getLastMarker()

```
app.project.sequences[index].markers.getLastMarker()
app.project.rootItem.children[index].getMarkers().getLastMarker()
```

Description

Retrieve the very last marker object, sorted by time in seconds, on a given project item or sequence.

Parameters

None.

Returns

Marker object or `undefined`.

25.2.5 MarkerCollection.getNextMarker()

```
app.project.sequences[index].markers.getNextMarker(currentMarker)
app.project.rootItem.children[index].getMarkers().getNextMarker(currentMarker)
```

Description

Get the next available marker, sorted by seconds, starting from a given one.

Parameters

Argument	Type	Description
`currentMarker`	*Marker object*	A starting marker object, from which to get a next one.

Returns

Marker object or `undefined`.

25.2.6 MarkerCollection.getPrevMarker()

```
app.project.sequences[index].markers.getPrevMarker(currentMarker)
app.project.rootItem.children[index].getMarkers().getPrevMarker(currentMarker)
```

Description

Get the previous available marker, sorted by seconds, starting from a given one.

Parameters

Argument	Type	Description
`currentMarker`	*Marker object*	A starting marker object, from which to get a previous one.

Returns

Marker object or `undefined`.

CHAPTER 26

ProjectCollection object

```
app.projects
app.production.projects
```

The ProjectCollection object represents a collection of *Project objects*.

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

26.1 Attributes

26.1.1 ProjectCollection.numProjects

```
app.projects.numProjects
app.production.projects.numProjects
```

Description

The total number of projects and productions found in the Project panel.

Type

Integer, read-only.

CHAPTER 27

ProjectItemCollection object

`app.project.rootItem.children`

The ProjectItemCollection object represents a collection of *ProjectItem objects* in an active project.

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

27.1 Attributes

27.1.1 ProjectItemCollection.numItems

`app.project.rootItem.children.numItems`

Description

The total number of items in the active project.

Type

Integer, read-only.

CHAPTER 28

SequenceCollection object

`app.project.sequences`

The SequenceCollection object represents a collection of all the *Sequence objects* in the active project.

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

28.1 Attributes

28.1.1 SequenceCollection.numSequences

`app.project.sequences.numSequences`

Description

The total number of sequences in the active project.

Type

Integer, read-only.

CHAPTER 29

TrackCollection object

```
app.project.sequences[index].audioTracks
app.project.sequences[index].videoTracks
```

The TrackCollection object represents a collection of *Track objects* in a sequence.

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

29.1 Attributes

29.1.1 TrackCollection.numTracks

```
app.project.sequences[index].audioTracks.numTracks
app.project.sequences[index].videoTracks.numTracks
```

Description

The total number of tracks in the sequence.

Type

Integer, read-only.

CHAPTER 30

TrackItemCollection object

```
app.project.sequences[index].audioTracks[index].clips
app.project.sequences[index].videoTracks[index].clips
```

The TrackItemCollection object represents a collection of *TrackItem objects* on a track.

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

30.1 Attributes

30.1.1 TrackItemCollection.numItems

```
app.project.sequences[index].audioTracks[index].clips.numItems
app.project.sequences[index].videoTracks[index].clips.numItems
```

Description

The total number of clips on a track.

Type

Integer, read-only.