JavaScript™ for Acrobat® 3D Annotations API Reference

Adobe® Acrobat® SDK

February 2011

Version 10.0

Adobe® Acrobat® X SDK JavaScript™ for Acrobat 3D Annotations API Reference for Microsoft® Windows® and Mac OS®

Edition 4.0, February 2011

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

Preface

The JavaScript™ API lets you manipulate 3D annotations within Adobe® PDF documents.

What's in this guide?

This document provides a brief overview of the API followed by a description of the objects.

Who should read this guide?

This guide is for developers who want to enhance the 3D experience of the user beyond the default behaviors. Using the JavaScript API for 3D annotations, you can specify the render modes and 3D matrix transformations of any of the individual meshes; set camera position, target, and field of view; detect mouse and keyboard events; control animations; and many more behaviors.

Related documentation

This document refers to the following sources for additional information about 3D annotations, JavaScript, and related technologies. The Adobe Acrobat® documentation is available through the Acrobat Developer Center.

Document	Description
Developing Acrobat Applications Using JavaScript	Using JavaScript to develop and enhance standard workflows in Acrobat and Adobe Reader®.
JavaScript for Acrobat API Reference	Detailed descriptions of JavaScript APIs for developing and enhancing workflows in Acrobat and Adobe Reader.
PDF Reference	A detailed description of the PDF file format.

1 JavaScript for Acrobat 3D Annotations API Reference

Introduction

To create 3D annotations and to attach scripts to them using this API, you need Adobe® Acrobat® Professional. Scripts attached to 3D annotations can run on Acrobat Pro Extended, Acrobat Pro, Acrobat Standard, and Adobe Reader® for Windows® and Mac OS® platforms. Unless otherwise noted, all JavaScript objects, properties, and methods have support starting in version 7.0.

The 3D JavaScript engine, which is distinct from the JavaScript engine for Acrobat, can be accessed in one of two ways. The primary way is by attaching a default script to the 3D annotation. This can be accomplished while placing a 3D annotation using the 3D Tool or on an existing 3D annotation by accessing its properties dialog box using the Select Object tool. This script will be run directly by the 3D JavaScript engine.

In addition, Acrobat provides a mechanism to directly access the entire 3D JavaScript engine API from within the Acrobat scripting engine by means of the JavaScript `Annot3D.context3D` property. For more information about JavaScript for Acrobat and its `Annot3D` object, see the JavaScript for Acrobat API Reference and Developing Acrobat Applications Using JavaScript.

The following example illustrates how to access the 3D JavaScript engine. In this example, a button (or link) contains JavaScript code that rotates the U3D object named "Axes".

```
// Get index of page containing the Annot3D object (count starts at 0).
pageIndex = this.pageNum;

// Index of the Annot3D (count starts at 0).
annotIndex = 0;

// Get a reference to the Annot3D script context.
c3d = this.getAnnots3D( pageIndex )[ annotIndex ].context3D;

// Get a reference to the node in the scene named "Axes".
axes = c3d.scene.nodes.getByName( "Axes" );

// Rotate the object about the X-Axis PI/6 radians (30 degrees).
axes.transform.rotateAboutXInPlace( Math.PI / 6 );
```

More extensive actions can be executed by having a button or link get the `SceneContext3d` object and call a function defined in the default script of the 3D annotation, as in the following example.

```
// Get the Annot3D script context of the targeted annotation.
context3D = getAnnots3D(0)[0].context3D;

// Call the JavaScript function setRenderMode() defined in the default
// script of the referenced 3D annotation.
context3D.setRenderMode("transparent");
```

The default script of the 3D annotation makes the definition.

```
function setRenderMode( renderModeName ) {
   for (var i=0; i < scene.meshes.count; i++) {
```

```
            scene.meshes.getByIndex(i).renderMode = renderModeName;
        }
    }
```

Object overview

This section provides an overview of the objects in the 3D JavaScript API.

Basic objects

There are several basic objects, such as `Color`, `Matrix4x4`, and `Vector3`, that are used to create general-purpose objects. The basic objects are used throughout the API and are only meaningful when attached to objects such as `Scene` or `Runtime`. For example, you could create a `Color` object and use it to set the `Background` color of a `Canvas`.

Vector3 Examples

```
v1 = new Vector3( 1.2, 3, 4.5 );
v2 = new Vector3( 5, 8, 13 );
v3 = new Vector3();
```

Matrix4x4 Examples

```
m1 = new Matrix4x4().rotateAboutX(Math.PI/1.5).rotateAboutY(Math.PI/3);
m2 = new Matrix4x4().rotateAboutZ(Math.PI/4).translate(new Vector3(0,5,0));
m3 = new Matrix4x4(m1);
```

Color Examples

```
c1 = new Color( 0.6, 0.8, 1.0 ); // light blue
c2 = new Color( 0.5, 0.5, 0.5 ); // middle grey
c3 = new Color(); //black

// A function to blend two Colors
Color.prototype.blend = function( color, amount )
{
   red       = ( this.r * ( 1 - amount ) ) + ( color.r * amount );
   green     = ( this.g * ( 1 - amount ) ) + ( color.g * amount );
   blue      = ( this.b * ( 1 - amount ) ) + ( color.b * amount );
   return( new Color( red, green, blue ) );
}
c4 = c1.blend( c2, 0.25 );
```

Scene object

The `Scene` is an object that contains all of the 3D-related content. It can be accessed using the global variable `scene`, which is a reference to the main `Scene` object. Most of the contents of the `Scene` are structured into a hierarchy of `Node` objects, and maintains lists of all these objects in the form of a `SceneObjectList`.

For more information, see Scene.

Canvas object

Represents a rectangular region into which a `Scene` is rendered from a particular viewpoint.

For more information, see Canvas.

Runtime object

The `Runtime` object is used to represent the instance of the playback engine. It manages all event processing and places where the graphic and textual content is rendered. It is accessed via the global variable `runtime`, which is a reference to the main `Runtime` object.

For more information, see Runtime.

Console object

The Console is the Acrobat text output area. It is helpful in debugging scripts.

Resource objects

Some objects, such as `Image`, are driven by content that is streamed from a file or over a network. To create an `Image`, load a .png, .jpg, or .gif file as a `Resource`, which you may subsequently use to create a new `Image` object, as shown in the following example:

```
faceRes = new Resource("pdf://picture.jpg");
faceImage = new Image( faceRes );
aMaterial = scene.meshes.getByIndex(0).material;
aMaterial.diffuseTexture.setImage( faceImage );
```

For more information, see Resource and Image.

Event handlers

There are several types of event handlers:

- `CameraEventHandler`
- `KeyEventHandler`
- `MouseEventHandler`
- `MenuEventHandler`
- `RenderEventHandler`
- `ScrollWheelEventHandler`
- `SelectionEventHandler`
- `TimeEventHandler`
- `ToolEventHandler`

Each one responds to a different type of event during simulation. They use a callback mechanism to run a function when an event occurs. The event is passed as an argument to the event handler's `onEvent` function so that it can be queried when the function runs. Event handlers are registered via the `addEventHandler` method, of the `Runtime` object.

CamaraEvent

A `CamaraEvent` is created when a View is selected.

For more information, see `CamaraEvent`.

KeyEvent

A `KeyEvent` is created when a key is pressed or released while the 3D `Canvas` is in focus. The following example illustrates how to handle a key event:

```
myKeyHandler           = new KeyEventHandler();
myKeyHandler.onEvent   = function( event )
{
   console.print( "Key pressed with code: " + event.characterCode );
}
runtime.addEventHandler( myKeyHandler );
```

For more information, see `KeyEvent`.

MenuEvent

A `MenuEvent` is created when a custom menu item is selected. To create a custom menu item on the context menu, invoke the `Runtime` object's `addCustomMenuItem` method, which allows a script to be attached to the item selection event.

For more information, see `MenuEvent`.

MouseEvent

A `MouseEvent` is created when the mouse is clicked on an active 3D `Canvas` or the cursor moves over an active 3D `Canvas`. The following syntax could be used to handle a mouse event:

```
myMouseHandler                = new MouseEventHandler();
myMouseHandler.onMouseDown    = true;
myMouseHandler.target         = scene.meshes.getByIndex(0);
myMouseHandler.onEvent        = function( event )
{
   console.print( "Mouse down at pixel " + event.mouseX );
   console.print( ", " + event.mouseY );
}
runtime.addEventHandler( myMouseHandler );
```

For more information, see `MouseEvent`.

RenderEvent

A `RenderEvent` is created immediately before an instance of the `Canvas` is drawn. If there is a split view in Acrobat resulting in two visible 3D rendered areas, a unique `RenderEvent` will be called for each of them. This is necessary in the case of a camera-aligned image (sprite) in the 3D content that needs to be pixel-aligned. Since the pixel dimensions of the two areas are possibly different, there are two callbacks that pass the different dimensions. This makes it possible to modify the `Scene` in the appropriate manner before it is drawn.

For more information, see `RenderEvent`.

ScrollWheelEvent

A `ScrollWheelEvent` object is created when the mouse scroll wheel is activated over an active 3D `Canvas` object.

For more information, see `ScrollWheelEvent`.

SelectionEvent

A `SelectionEvent` object is created when an object is selected from an active 3D `Canvas` object or from a model tree. If the selection is made from a `Canvas` object, a `MouseEvent` is also created.

For more information, see `SelectionEvent`.

TimeEvent

A `TimeEvent` is created when the 3D annotation is enabled and simulation is active. The time and deltaTime properties are measured in terms of simulation time, not real time. `TimeEvent` objects are used to drive animation. If you need an accurate, real-time measurement, use the JavaScript `Date` object. The following syntax is used to handle a time event:

```
myTimeHandler               = new TimeEventHandler();
myTimeHandler.onEvent       = function( event )
{
   console.print( "Current simulation time is:" + event.time );
   console.print( " second(s)" );
}
runtime.addEventHandler( myTimeHandler );
```

For more information, see `TimeEvent`.

ToolEvent

A `ToolEvent` is created when a tool is clicked in the Acrobat 3D toolbar. The `Runtime` object's `addCustomToolButton` method allows you to add a custom tool to the toolbar which will also be generated, and allows a script to be attached to the tool selection event.

For more information, see ToolEvent.

2 JavaScript Objects for Acrobat 3D

This chapter describes the following 3D JavaScript objects:

Note: A property labeled as R (read-only) is one whose value cannot be set. An object labeled as R (read-only) is one whose reference cannot be modified, though the object itself can be set and its properties may be modified. Unless otherwise indicated, all properties and objects labeled with R/W have read/write access.

Animation

A type of SceneObject, used to store keyframe animation sequences of Node objects in the Scene. In addition to the methods and properties below, it also contains the same methods and properties as a SceneObject.

Properties

Property	Type	Access	Description
currentTime	number	R/W	The current time measured in seconds.
endTime	number	R	The end time of the sequence, measured in seconds.
framesPerSecond	number	R	The number of frames per second used to author the sequence.
length	number	R	The length of the Animation, measured in seconds.
startTime	number	R	The start time of the sequence, measured in seconds.

Background

Represents the background of a `Canvas`. It can be used as a target of a `MouseEventHandler`. (See `Canvas` and `MouseEventHandler`.)

Properties

Property	Type	Access	Description
`image`	Image	R/W	Acrobat 7.0.7 The `Image` to be used by the `Background`.
`FlashMovie`	FlashMovie	R/W	Acrobat 9.0 The `FlashMovie` to be used by the `Background`. `FlashMovie` replaces any `Image` or `Color` currently being used by the `Background`

getColor

Obtains the background `Color`.

Syntax

```
getColor()
```

Returns

A `Color` object representing the background color of the `Canvas`.

getImage

Deprecated

Obtains the background `Image`.

Syntax

```
getImage()
```

Returns

An `Image` object representing the background image of the `Canvas`.

setColor

Sets the background `Color`. If only one color is passed to this method, the background is a constant color. If two colors are passed to this method, the background is a linear gradient from top to bottom, with the first color argument representing the top color and the second representing the bottom color.

Syntax

```
setColor(topColor, bottomColor)
```

Parameters

`topColor`	A `Color` object representing the desired background color. If `bottomColor` is used, `topColor` represents the top background color used in a linear gradient.
`bottomColor`	(Optional) A `Color` object representing the bottom background color used in a linear gradient.

Returns

undefined

setImage

Deprecated

Sets the background Image.

Syntax

```
setImage(image)
```

Parameters

`image`	An `Image` object representing the desired background image.

Returns

undefined

Bone

A type of `Node` used to modify the shape of a `Mesh`, and is usually moved over time to create animated characters. It contains the same methods and properties as a `Node`.

Related objects are `Node` and `Mesh`.

BoundingBox

Represents an axis-aligned bounding box.

Properties

Property	Type	Access	Description
`center`	Vector3	R	Acrobat 7.0.7 The coordinates of the `BoundingBox` center.
`max`	Vector3	R	The coordinates of the `BoundingBox` corner with the greatest x, y, and z values.
`min`	Vector3	R	The coordinates of the `BoundingBox` corner with the smallest x, y, and z values.

Camera

A `Node` that controls the projection from world space to screen space. In addition to the methods and properties below, it also contains the same methods and properties as a `Node`.

Properties

Property	Type	Access	Description
`absoluteBindingScale`	number	R/W	Acrobat 7.0 The absolute binding scale value for the camera.
`binding`	string	R/W	The view plane calculation type, which can take one of the following values: • `"min"` • `"max"` • `"horizontal"` • `"vertical"`
`BINDING_HORIZONTAL`	string	R	Acrobat 7.0.7 A string constant for the binding value of `"horizontal"`.
`BINDING_MAX`	string	R	Acrobat 7.0.7 A string constant for the binding value of `"max"`.
`BINDING_MIN`	string	R	Acrobat 7.0.7 A string constant for the binding value of `"min"`.
`BINDING_VERTICAL`	string	R	Acrobat 7.0.7 A string constant for the binding value of `"vertical"`.
`far`	number	R/W	The distance from the `Camera` to the far clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`fov`	number	R/W	The size of the field of view for perspective `Camera` objects, measured in radians.
`near`	number	R/W	The distance from the `Camera` to the near clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`position`	Vector3	R	The position of the origin of the `Camera` in world space.
`positionLocal`	Vector3	R	The position of the origin of the `Camera` in local space.

Property	Type	Access	Description
projectionType	string	R/W	The type of projection, which can take one of the following values: • "perspective" • "orthographic"
roll	number	R/W	The roll angle of the Camera, measured in radians.
target	Node	R	The current Node used as the Camera object's target.
targetPosition	Vector3	R	The position of the Camera object's target in world space.
targetPositionLocal	Vector3	R/W	The position of the Camera object's target in local space.
TYPE_ORTHOGRAPHIC	string	R	Acrobat 7.0.7 A string constant for the camera projection type of "orthographic".
TYPE_PERSPECTIVE	string	R	Acrobat 7.0.7 A string constant for the camera projection type of "perspective".
up	Vector3	R	The up direction in world space.
upLocal	Vector3	R	The up direction in local space.
useAbsoluteBinding	Boolean	R	Acrobat 7.0 Determines whether the camera uses absolute binding for its projection.
viewPlaneSize	number	R/W	The size of the view plane for orthographic Camera objects, measured in scene units.

getScreenFromPosition

Obtains the screen coordinates of the provided 3D position.

Syntax

```
getScreenFromPosition(position, canvasWidth, canvasHeight)
```

Parameters

position	A Vector3 object representing the 3D position.

canvasWidth	The width of the Canvas, measured in pixels.
canvasHeight	The height of the Canvas, measured in pixels.

Returns

A Vector3 object representing the screen coordinates, with x and y as pixel positions and z equal to zero.

See Vector3 for more information on the return object.

getDirectionFromScreen

Obtains the direction from the normalized coordinates

Syntax

```
getDirectionFromScreen(x, y, canvasWidth, canvasHeight)
```

Parameters

x	The x-coordinate, measured in pixels.
y	The y-coordinate, measured in pixels.
canvasWidth	The width of the Canvas, measured in pixels.
canvasHeight	The height of the Canvas, measured in pixels.

Returns

A Vector3 object representing the direction.

See Vector3 for more information on the return object.

CamaraEvent

Describes the format of the object that is passed as an argument to the `onEvent` method of the `CameraEventHandler` object.

Properties

Property	Type	Access	Description
`binding`	string	R	The view plane calculation type, which can take one of the following values: • "min" • "max" • "horizontal" • "vertical"
`canvas`	Canvas	R	The `Canvas` in which the event took place.
`currentTool`	string	R	The name of the current tool.
`far`	number	R	The distance from the `Camera` to the far clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`fov`	number	R	The size of the field of view for perspective `Camera` objects, measured in radians.
`isNewCanvas`	Boolean	R	Deprecated Determines whether this is the first event for this `Canvas`.
`near`	number	R	The distance from the `Camera` to the near clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`projectionType`	string	R	The type of projection, which can take one of the following values: • "perspective" • "orthographic"
`targetDistance`	number	R	The distance from the `Camera` to its target.
`transform`	Matrix4x4	R	The Camera object's transformation matrix.
`viewPlaneSize`	number	R	The size of the view plane, measured in scene units.

CameraEventHandler

Exposes a callback mechanism that allows a function to be evaluated when an camera event occurs. Event handlers are registered with the Runtime `addEventHandler` method.

CameraEventHandler

A constructor that creates a new `CameraEventHandler` object.

Syntax

```
new CameraEventHandler()
```

Returns

A `CameraEventHandler` object.

onEvent

A method that is called when a view is selected from the list of views on the 3D toolbar or in the context menu for an active 3D annotation.

syntax

```
onEvent(event)
```

Parameters

`event`	A `CameraEvent` object representing the event.

Returns

undefined

Canvas

Represents a rectangular region into which the `Scene` is rendered from the viewpoint of the attached `Camera`.

See related objects, `Scene` and `Camera`.

Properties

Property	Type	Access	Description
`background`	Background	R	The `Background` object associated with the `Canvas`.

getCamera

Obtains the `Camera` object attached to the `Canvas`.

Syntax

```
getCamera()
```

Returns

A `Camera` object.

setCamera

Sets the `Camera` object attached to the `Canvas`.

Syntax

```
setCamera(camera)
```

Parameters

`camera`	The `Camera` object used to set the object's value.

Returns

undefined

ClippingPlane

An object representing a plane, within the `Scene`, that clips all geometry on one side of it. It is created by invoking the `createClippingPlane` method of the `Scene` object.

remove

Removes the `ClippingPlane` object from the `Scene`.

Syntax

```
remove()
```

Returns

undefined

Color

An object that represents a RGB encoded color.

Properties

Property	Type	Description
b	number	The blue component, which can be a value from 0.0 to 1.0.
g	number	The green component, which can be a value from 0.0 to 1.0.
r	number	The red component, which can be a value from 0.0 to 1.0.

Color

A constructor that creates a Color object, initialized to black.

Syntax

```
new Color()
```

Returns

A Color object, initialized to black.

Color

A constructor that creates a Color object, initialized to the supplied RGB values.

Syntax

```
new Color(r, g, b)
```

Parameters

r	The red component, which can be a value from 0.0 to 1.0.
g	The green component, which can be a value from 0.0 to 1.0.
b	The blue component, which can be a value from 0.0 to 1.0.

Returns

A Color object, initialized to the supplied RGB values.

set

Sets the Color object's value using an existing Color object

Syntax

```
set(color)
```

Parameters

color	The Color object used to set the object's value.

Returns

undefined

set

Acrobat 7.0.7

Sets the Color object's value using the given RGB components.

Syntax

```
set(r, g, b)
```

Parameters

r	The red component, which can be a value from 0.0 to 1.0.
g	The green component, which can be a value from 0.0 to 1.0.
b	The blue component, which can be a value from 0.0 to 1.0.

Returns

undefined

set3

Deprecated

Sets the Color object's value using the given RGB components.

Syntax

```
set3(r, g, b)
```

Parameters

r	The red component, which can be a value from 0.0 to 1.0.
g	The green component, which can be a value from 0.0 to 1.0.
b	The blue component, which can be a value from 0.0 to 1.0.

Returns

undefined

Console

This object can direct output to the JavaScript console in Acrobat for debugging purposes. The variable `console` is a global reference to this object.

print

Prints a string to the JavaScript Console.

Syntax

```
print(string)
```

Parameters

`string`	The text to be printed to the JavaScript Console.

Returns

undefined

println

Prints a string with an accompanying newline to the JavaScript Console.

Syntax

```
println(string)
```

Parameters

`string`	The text to be printed to the JavaScript Console.

Returns

undefined

Dummy

Deprecated

A `Node` object used as an empty placeholder or a group within a `Scene`.

FlashEvent

Acrobat 9.0

An object that is passed as an argument to the `onEvent` method of the FlashEventHandler object.

Properties

Property	Type	Access	Description
`command`	string	R	For a `FlashEvent` of type `"command"`, this is the string representation of the command that has been sent through the ActionScript `FSCommand` function or through the `ExternalInterface.call` method. To execute the command, run the JavaScript function `eval` with the command string as an argument.
`target`	`FlashMovie`	R	The target `FlashMovie` that the `FlashEvent` is from.
`type`	string	R	The type of `FlashEvent`, which can be `"command"`, `"progress"`, or `"stateChange"`.
`TYPE_COMMAND`	string	R	A string constant for the `FlashEvent` type of `"command"`.
`TYPE_PROGRESS`	string	R	A string constant for the `FlashEvent` type of `"progress"`.
`TYPE_STATECHANGE`	string	R	A string constant for the `FlashEvent` type of `"stateChange"`.
`value`	integer	R	The value for the corresponding type of `FlashEvent`. The interpretation of `value` depends on the event type, `"progress"` or `"stateChange"`. **`"progress"`**: `value` is an integer from 0 to 100 representing the load progress of the `FlashMovie`. **`"stateChange"`**: `value` is an integer signifying the ready state of the `FlashMovie`. Permitted values are `0` (Loading), `1` (Uninitialized), `2` (Loaded), `3` (Interactive), `4` (Complete).

FlashEventHandler

Acrobat 9.0

An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs in a `FlashMovie` object. Event handlers are registered with the Runtime.`addEventHandler` method.

Properties

Property	Type	Access	Description
`target`	`FlashMovie`	R/W	When set, the `FlashEventHandler` will only report FlashEvents from the provided target `FlashMovie`.

onEvent

A method that is called when an `ExternalInterface.call` method or `MMExecute` command is invoked from the `FlashMovie`'s ActionScript.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `FlashEvent` object representing the event.

Returns

undefined

FlashEventHandler

The constructor that creates a new `FlashEventHandler`.

Syntax

```
new FlashEventHandler()
```

Returns

A `FlashEventHandler` object.

FlashMovie

Acrobat 9.0

An object that represents a Flash movie in the `Scene`.

Properties

Property	Type	Access	Description
`alignMode`	integer	R/W	A bit flag that sets the alignment of the movie within the Scene. Values are +1 (left aligned), +2 (right aligned), +4 (top aligned), and +8 (bottom aligned).
`ALIGN_MODE_BOTTOM`	string	R	A string constant for the `FlashMovie scaleMode` type of `"bottom"`.
`ALIGN_MODE_LEFT`	string	R	A string constant for the `FlashMovie scaleMode` type of `"left"`.
`ALIGN_MODE_RIGHT`	string	R	A string constant for the `FlashMovie scaleMode` type of `"right"`.
`ALIGN_MODE_TOP`	string	R	A string constant for the `FlashMovie scaleMode` type of `"top"`.
`backgroundColor`	integer	R/W	Override the background color of a movie. An integer of the form (*red* * 65536 + *green* * 256 + *blue*). Use a value of -1 for the default movie color. The values for *red*, *green* and *blue* are integers between 0 and 255, inclusive, and represent the color components of red, green, and blue, respectively, in the RGB color model.
`desiredResolutionX`	integer	R/W	The desired resolution width for the `FlashMovie` content to be rendered at.
`desiredResolutionY`	integer	R/W	The desired resolution height for the `FlashMovie` content to be rendered at.
`frameNum`	integer	R/W	The frame number of the currently displayed frame of the movie. Setting this property advances or rewinds the movie.
`hitEnabled`	Boolean	R/W	Determines whether mouse events travel through the `FlashMovie` to elements in the scene behind it. If `true`, mouse events are trapped.

Property	Type	Access	Description
`id`	integer	R	A unique ID for each `FlashMovie` in the scene.
`loop`	Boolean	R/W	A flag that determines whether the animation loops. If `true`, the animation loops. If `false`, it plays only once.
`opacity`	number	R/W	The opacity of the `FlashMovie` represented by a value from 0.0 to 1.0, where 1.0 is completely opaque.
`percentLoaded`	integer	R	The percent of the Adobe Flash Player movie that has streamed into the browser so far with possible values from from 0 to 100.
`playing`	Boolean	R	A flag that detects whether the movie is currently playing. If `true`, it is playing. If `false`, it is paused.
`quality`	integer	R/W	The current rendering quality. Permitted values are `0` (Low), `1` (High), `2` (AutoLow), and `3` (AutoHigh).
`readyState`	integer	R	The state of the `FlashMovie`. Permitted values are `0` (Loading), `1` (Uninitialized), `2` (Loaded), `3` (Interactive), `4` (Complete).
`resolutionType`	string	R/W	A string value that specifies the type of resolution to be used for the movie. Recognized values are `"custom"`, `"movie"`, and `"window"`.
`RESOLUTION_TYPE_CUSTOM`	string	R	A string constant for the `FlashMovie` resolution type of `"custom"`.
`RESOLUTION_TYPE_MOVIE`	string	R	A string constant for the `FlashMovie` resolution type of `"movie"`.
`RESOLUTION_TYPE_WINDOW`	string	R	A string constant for the `FlashMovie` resolution type of `"window"`.
`scaleMode`	string	R/W	The scale mode of the movie. The value of this property may be `"exact fit"`, `"no border"`, or `"show all"`.
`SCALE_MODE_EXACT_FIT`	string	R	A string constant for the `FlashMovie` `scaleMode` type of `"exact fit"`.
`SCALE_MODE_NO_BORDER`	string	R	A string constant for the `FlashMovie`'s `scaleMode` type of `"no border"`.

Property	Type	Access	Description
SCALE_MODE_SHOW_ALL	string	R	A string constant for the FlashMovie scaleMode type of "show all".
totalFrames	integer	R	The total number of frames in the movie. This is not available until the movie has loaded. Wait for ReadyState = 4.
x	integer	R/W	The x-position of the FlashMovie in the Canvas. Applies only to a FlashMovie if it is attached to the Foreground.
y	integer	R/W	The y-position of the FlashMovie in the Canvas. Applies only to a FlashMovie if it is attached to the Foreground.

FlashMovie

Creates a new FlashMovie from a Resource of type "flash".

Syntax

```
FlashMovie(FlashMovieResource)
```

Parameters

FlashMovieResource	A Resource of type "flash".

Returns

A FlashMovie object.

call

Calls into ActionScript with the ExternalInterface calling convention to an exposed method (ExternalInterface.addCallback in ActionScript). The call method returns the return value of the method specified as the first parameter.

Note: The JavaScript for Acrobat API Reference has the callAS method of the AnnotRichMedia object that uses the same mechanism as the call method.

Syntax

```
call(functionName, [argument1[, ...,argumentn]])
```

Parameters

`functionName`	A string representing the function name to call in the `FlashMovie` ActionScript engine.
`argument1,argument2, ...,argumentn`	A comma-delimited list of arguments of varying type to be passed to the function in ActionScript.

Returns

The return value from the called function, which can be of any type.

getVariable

A method that returns the value of the Flash variable specified by `varName`, and returns `undefined` if the variable does not exist.

Syntax

```
getVariable(varName)
```

Parameters

`varName`	A string representing the variable requested.

Returns

A string representing the value of the specified Flash variable, or `undefined`.

gotoFrame

Activates the frame number specified by `frameNumber` in the current movie. If the data for a requested frame is not yet available, the player goes to the last frame available and stops, causing unexpected results during playback. Use the `percentLoaded` property to determine if enough of the movie is available to execute the `gotoFrame()` method. The argument `frameNumber` is zero-based; that is, `frameNumber` is 0 in the first frame of the movie, 1 for the second frame, and so on. This differs from the `Goto` action within Flash, which begins at 1.

Syntax

```
gotoFrame(frameNumber)
```

Parameters

`frameNumber`	An integer representing the frame number.

Returns

undefined

isPlaying

A method that returns `true` if the movie is currently playing.

Syntax

```
isPlaying()
```

Returns

A Boolean type, `true` if the movie is playing, `false` otherwise.

pan

This method pans a zoomed-in movie to the coordinates specified by `x` and `y`. Use `mode` to specify whether the values for `x` and `y` are pixels or a percentage of the window. The `pan` method does not pan beyond the boundaries of the zoomed-in movie.

Syntax

```
pan(x, y, mode)
```

Parameters

`x`	An integer representing the x coordinate.
`y`	An integer representing the y coordinate.
`mode`	When `mode` is 0, the coordinates are pixels; when `mode` is 1, the coordinates are a percentage of the window.

Returns

undefined

play

Starts playing the movie.

Syntax

```
play()
```

Returns

undefined

rewind

Goes to the first frame.

Syntax

```
rewind()
```

Returns

undefined

setVariable

Sets the value of the Flash variable specified by `variableName` to the value specified by `value`.

Syntax

```
setVariable(varName, value)
```

Parameters

`varName`	A string representing the variable requested.
`value`	A string value to be set for the provided variable name.

Returns

undefined

setZoomRect

Zooms in on a rectangular area of the movie. The units of the coordinates are measured in twips (1440 units per inch).

Note: To calculate the dimensions of a rectangle in the correct units, set the ruler units to Points and multiply the coordinates by 20 to get twips. (There are 72 points per inch.)

Syntax

```
setZoomRect(left, top, right, bottom)
```

Parameters

`left`	An integer representing the left side of the rectangle.
`top`	An integer representing the top side of the rectangle.
`right`	An integer representing the right side of the rectangle.
`bottom`	An integer representing the bottom side of the rectangle.

Returns

undefined

stop

Stops playing the movie.

Syntax

```
stop()
```

Returns

undefined

zoom

This method zooms the view by a relative scale factor specified by percentage. For example, `zoom(50)` doubles the size of the objects in the view, `zoom(200)` reduces the size of objects in the view by one half, and `zoom(0)` resets the view to 100%. You cannot specify a scale factor that will zoom-out the original content further than 100%.

Syntax

```
zoom(percentage)
```

Parameters

`percentage`	An integer representing the zoom factor.

Returns

undefined

HitInfo

The object returned when a hit test occurs during a `MouseEvent`.

Properties

Property	Type	Access	Description
`distance`	number	R	The distance from the `Camera` to the `HitInfo` object's `position`.
`material`	material	R	Acrobat 8.1 The material of the node that was hit.
`position`	vector3	R	The position of the point where the hit occurred.
`surfaceNormal`	vector3	R	Acrobat 8.1 The normal direction at the hit location on the world-space surface.
`target`	node	R	The target of the hit test.
`textureCoordinate`	vector3	R	Acrobat 8.1 The texture coordinate of the material that was hit.

Host

Acrobat 7.0.7

An object that provides access to the JavaScript engine context and to pertinent objects within it. The variable `host` is a global reference to this object. It is a reference to the JavaScript Doc object in which the 3D annotation is contained.

Image

An object that represents an image.

Properties

Property	Type	Access	Description
`height`	number	R	The image's height, measured in pixels.
`width`	number	R	The image's width, measured in pixels.

Image

A constructor that creates an new `Image` object.

Syntax

```
new Image(resource)
```

Parameters

`resource`	An `Image` object used to create the new object.

Returns

An `Image` object.

See `Image` for more information about the return object.

KeyEvent

An object that is passed as an argument to the `onEvent` method of the `KeyEventHandler` object.

Properties

Property	Type	Access	Description
`canvas`	canvas	R	The `Canvas` in which the `KeyEvent` took place.
`canvasPixelHeight`	integer	R	The height, measured in pixels, of the `Canvas`.
`canvasPixelWidth`	integer	R	The width, measured in pixels, of the `Canvas`.
`characterCode`	integer	R	The value of the character pressed according to Acrobat's character mapping, as per this listing of Acrobat character codes:

#	Keys	#	Keys	#	Keys
		65	A	97	a
		66	B	98	b
		67	C	99	c
		68	D	100	d
28	Left	69	E	101	e
29	Right	70	F	102	f
30	Down	71	G	103	g
31	Up	72	H	104	h
		73	I	105	i
		74	J	106	j
32	Space	75	K	107	k
		76	L	108	l
		77	M	109	m
48	0	78	N	110	n
49	1	79	O	111	o
50	2	80	P	112	p
51	3	81	Q	113	q
52	4	82	R	114	r
53	5	83	S	115	s
54	6	84	T	116	t
55	7	85	U	117	u
56	8	86	V	118	v
57	9	87	W	119	w
		88	X	120	x
		89	Y	121	y
		90	Z	122	z

Property	Type	Access	Description
`ctrlKeyDown`	Boolean	R	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed. **Note:** Acrobat intercepts many of the Ctrl + key events because they are used for accelerators in the main application.
`currentTool`	string	R	The name of the current tool.
`shiftKeyDown`	Boolean	R	Determines whether the Shift key was pressed. **Note:** Holding down the shift key changes the value of the `KeyEvent.characterCode` property.

KeyEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when a key event occurs. Event handlers are registered with the Runtime `addEventHandler` method.

KeyEventHandler

A constructor that creates a new `KeyEventHandler` object.

Syntax

```
new KeyEventHandler()
```

Returns

A `KeyEventHandler` object.

onEvent

A method that is called when a key is pressed.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `KeyEvent` object representing the event.

Returns

undefined

Light

A `Node` object that illuminates meshes in the `Scene`.

There are different types of `Light` objects, each with their own distinct behavior. Infinite `Light` objects behave much like sunlight in that they cast parallel light in a given direction. Spot `Light` objects have a fixed cone angle that limits their beam to a conical projection. Point `Light` objects act similarly to a light bulb, where the light comes from a specific location in 3D space. Currently, none of the `Light` objects cast shadows.

In addition to the methods and properties that follow, the `Light` object also contains the same methods and properties as a `Node`.

Properties

Property	Type	Access	Description
`attenuationA`	number	R/W	The `a` coefficient for `attenuationType "abc"`.
`attenuationB`	number	R/W	The `b` coefficient for `attenuationType "abc"`.
`attenuationC`	number	R/W	The `c` coefficient for `attenuationType "abc"`.
`attenuationType`	string	R/W	The style of attenuation for the `Light` object being represented. Attenuation determines how fast the light intensity decreases with distance. The attenuation type of `"abc"` uses the equation 1 / max((a + bd + cdd), 1) to determine the intensity where d is the distance from the light. One of the following values may be assigned: • `"abc"`. • `"none"`
`ATTENUATION_ABC`	string	R	Acrobat 7.0.7 A string constant for the `attenuationType` of `"abc"`.
`ATTENUATION_NONE`	string	R	Acrobat 7.0.7 A string constant for the `attenuationType` of `"none"`.
`brightness`	number	R/W	Specifies the brightness of the emission from the `Light`. A value of `1` represents a brightness of 100%, though the property may be assigned higher values.
`color`	Color	R	Specifies the color of the light.

Property	Type	Access	Description
direction	Vector3	R	The direction toward which the light is pointing.
directionLocal	Vector3	R	Acrobat 7, but not documented until Acrobat 8.1 The direction toward which the light is pointing relative to its parent Node.
innerConeAngle	number	R/W	The angle, measured in radians, about the direction in which the light is of uniform full density.
innerRadius	number	R/W	The distance within which the light is of uniform full density.
outerConeAngle	number	R/W	The angle, measured in radians, about the direction outside of which the light's intensity is zero.
outerRadius	number	R/W	The distance beyond which the light's intensity is zero.
position	Vector3	R	The position of the Light object.
positionLocal	Vector3	R	The position of the Light object relative to its parent Node .
type	string	R/W	The type of Light object being represented. One of the following values may be assigned: • "point" • "spot" • "infinite"
TYPE_INFINITE	string	R	Acrobat 7.0.7 A string constant for the Light type of "infinite".
TYPE_POINT	string	R	Acrobat 7.0.7 A string constant for the Light type of "point".
TYPE_SPOT	string	R	Acrobat 7.0.7 A string constant for the Light type of "spot".

Material

A `SceneObject` that controls the appearance of materials using the fixed function shader. In addition to the properties below, it contains the same methods and properties as a `SceneObject`.

Properties

Property	Type	Access	Description
`ambientColor`	Color	R	The ambient color.
`ambientTexture`	Texture	R	The ambient texture.
`bumpTexture`	Texture	R	A texture map whose value is used to describe the roughness of the object.
`diffuseColor`	Color	R	The matte color of an object.
`diffuseTexture`	Texture	R	A texture map that is used for the matte color of the object.
`emissiveColor`	Color	R	Emissive color except that it is does not require that any lighting to display. An object with an emissive color of white and no texture will appear pure white in the scene.
`emissiveTexture`	Texture	R	The emissive texture. Emissive texture is similar to ambient color, except that it is does not require that any lighting to display. An object with an emissive color of white and no texture will appear pure white in the scene.
`opacity`	number	R/W	The total opacity of the material.
`opacityTexture`	Texture	R	A texture map whose brightness is used for the level of opacity of the object. White signifies completely opaque while black signifies completely transparent.
`phongExponent`	number	R/W	The Phong exponent. The Phong exponent defines the “tightness” of the highlight. A higher exponent results in a smaller, tighter highlight while a lower exponent results in a broader flatter one.
`reflectionStrength`	number	R/W	The reflection level, which can be a value from `0.0` to `1.0`.
`reflectionTexture`	Texture	R	The reflection texture, also known as an environment map, the texture is used to store the image of the environment surrounding the rendered object. It simulates the reflection of a mirrored surface

Property	Type	Access	Description
specularColor	Color	R	The specular color. The specularColor is the color of the highlight on the material.
specularStrength	number	R/W	The specular strength, which is a measure of how shiny the material is.

attachFlashMovie

Acrobat 9.0

Sets the provided FlashMovie as the diffuse texture for the Material.

Syntax

```
attachFlashMovie(movie)
```

Parameters

movie	The FlashMovie object to be used as the diffuse texture.

Returns

undefined

Matrix4x4

A four-by-four matrix commonly used for transformations.

Properties

Property	Type	Access	Description
`determinant`	number	R/W	The determinant of the matrix.
`inverse`	Matrix4x4	R	The inverse of the matrix.
`scaleComponent`	Vector3	R	The scale component of the transformation.
`translation`	Vector3	R	The translation component of the transformation.
`transpose`	Matrix4x4	R	The transpose of the matrix.

Matrix4x4

A constructor that creates a new `Matrix4x4` object initialized to the identity matrix.

Syntax

```
new Matrix4x4()
```

Returns

A `Matrix4x4` object initialized to the identity matrix.

Matrix4x4

A constructor that creates a new `Matrix4x4` object initialized to the specified matrix.

Syntax

```
new Matrix4x4(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used to initialize the new matrix.

Returns

A `Matrix4x4` object initialized to the specified matrix.

invertInPlace

Inverts the matrix.

Returns

undefined

isEqual

Determines whether the current matrix is equal to the specified matrix.

Syntax

```
isEqual(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used for the comparison.

Returns

Returns `true` if the matrices are equal, `false` otherwise.

multiply

Multiplies the current matrix by the specified matrix.

Syntax

```
multiply(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used for the multiplication.

Returns

A `Matrix4x4` object.

multiplyInPlace

Multiplies the current matrix by the specified matrix, and updates the current matrix with the resulting value.

Syntax

```
multiplyInPlace(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used for the multiplication.

Returns

undefined

rotateWithQuaternion

Rotates the current matrix using the specified `Quaternion`.

Syntax

```
rotateWithQuaternion(quaternion)
```

Parameters

`quaternion`	A `Quaternion` object used for the rotation.

Returns

A `Matrix4x4` object.

rotateWithQuaternionInPlace

Rotates the current matrix using the specified quaternion, and updates the current matrix with the resulting value.

Syntax

```
rotateWithQuaternionInPlace(quaternion)
```

Parameters

`quaternion`	A `Quaternion` object used for the rotation.

Returns

undefined

rotateAboutLine

Rotates the current matrix about the specified line.

Syntax

```
rotateAboutLine(angle, start, end)
```

Parameters

`angle`	The angle of rotation, in radians.
`start`	A point described by a `Vector3` object used to specify the beginning of the line of rotation, which is represented by `start - end`.
`end`	A point described by a `Vector3` object used to specify the end of the line of rotation, which is represented by `start - end`.

Returns

A `Matrix4x4` object.

rotateAboutLineInPlace

Rotates the current matrix about the specified line, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutLineInPlace(angle, start, end)
```

Parameters

`angle`	The angle of rotation, in radians.
`start`	A `Vector3` object used to specify the line of rotation, which is represented by `start - end`.
`end`	A `Vector3` object used to specify the line of rotation, which is represented by `start - end`.

Returns

undefined

rotateAboutX

Rotates the current matrix about the x axis.

Syntax

```
rotateAboutX(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

A `Matrix4x4` object.

rotateAboutXInPlace

Rotates the current matrix about the x axis, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutXInPlace(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

undefined

rotateAboutVector

Rotates the current matrix about the specified vector.

Syntax

```
rotateAboutVector(angle, axis)
```

Parameters

`angle`	The angle of rotation, in radians.
`axis`	A `Vector3` object about which the matrix is rotated.

Returns

A `Matrix4x4` object.

rotateAboutVectorInPlace

Rotates the current matrix about the specified vector, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutVectorInPlace(angle, axis)
```

Parameters

`angle`	The angle of rotation, in radians.
`axis`	A `Vector3` object about which the matrix is rotated.

Returns

undefined

rotateAboutY

Rotates the current matrix about the y axis.

Syntax

```
rotateAboutY(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

A `Matrix4x4` object.

rotateAboutYInPlace

Rotates the current matrix about the y axis, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutYInPlace(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

undefined

rotateAboutZ

Rotates the current matrix about the z axis.

Syntax

```
rotateAboutZ(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

A `Matrix4x4` object.

rotateAboutZInPlace

Rotates the current matrix about the z axis, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutZInPlace(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

undefined

scale

Scales the current matrix using the specified scaling components.

Syntax

```
scale(x, y, z)
```

Parameters

`x`	The scaling component in the x direction.
`y`	The scaling component in the y direction.
`z`	The scaling component in the z direction.

Returns

A `Matrix4x4` object.

scaleInPlace

Scales the current matrix using the specified scaling components, and updates the current matrix with the resulting value.

Syntax

```
scaleInPlace(x, y, z)
```

Parameters

x	The scaling component in the x direction.
y	The scaling component in the y direction.
z	The scaling component in the z direction.

Returns

undefined

set

Sets the value of the current matrix using the specified matrix.

Syntax

```
set(matrix)
```

Parameters

matrix	The matrix whose value is copied into the current matrix.

Returns

undefined

set

Acrobat 8.1

Sets the value of the current matrix using an array.

Syntax

```
set( array )
```

Parameters

`array`	The array of length 16 whose values are copied into the current matrix.

Returns

undefined

set

Acrobat 8.1

Sets the value of the current matrix using 16 numeric values.

Syntax

```
set(v0, v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15)
```

Parameters

`v0-v15`	Number values for the given indices of the matrix.

Returns

undefined

setIdentity

Sets the value of the current matrix to the identity matrix.

Syntax

```
setIdentity()
```

Returns

undefined

setView

Sets the current matrix according to the specified component vectors.

Syntax

```
setView(position, direction, up)
```

Parameters

`position`	A `Vector3` object used to specify the position component.
`direction`	A `Vector3` object used to specify the direction component.
`up`	A `Vector3` object used to specify the upward component.

Returns

undefined

transformDirection

Transforms the specified vector by the current matrix.

Syntax

```
transformDirection(vector)
```

Parameters

`vector`	The `Vector3` object to be transformed.

Returns

A `Vector3` object.

transformPosition

Transforms the specified position by the current matrix.

Syntax

```
transformPosition(position)
```

Parameters

`position`	A `Vector3` object representing the position to be transformed.

Returns

A `Vector3` object.

translate

Translates the current matrix by the components of the specified vector.

Syntax

```
translate(translation)
```

Parameters

`translation`	The `Vector3` object whose components are used to perform the matrix translation.

Returns

A `Matrix4x4` object.

translateInPlace

Translates the current matrix by the components of the specified vector, and updates the current matrix with the resulting value.

Syntax

```
translateInPlace(translation)
```

Parameters

`translation`	The `Vector3` object whose components are used to perform the matrix translation.

Returns

undefined

transposeInPlace

Sets the value of the current matrix to its transpose.

Syntax

```
transposeInPlace()
```

Returns

undefined

MenuEvent

An object that is passed as an argument to the `onEvent` method of the `MenuEventHandler` object.

Properties

Property	Type	Access	Description
`canvas`	Canvas	R	The `Canvas` in which the `MenuEvent` took place.
`currentTool`	string	R	The name of the current tool.
`menuItemChecked`	Boolean	R	Determines whether the menu item was selected.
`menuItemName`	string	R	The name of the selected menu item.

MenuEventHandler

A `MenuEventHandler` object exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime` `addEventHandler` method.

MenuEventHandler

A constructor that creates a new `MenuEventHandler` object.

Syntax

```
new MenuEventHandler()
```

Returns

A `MenuEventHandler` object.

onEvent

A method that is called when a custom menu item is selected on the context menu for an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `MenuEvent` object representing the event.

Returns

undefined

Mesh

A `Node` object that contains geometry. A `Mesh` object with no geometry has children `Node` objects that can be transformed as a group. In addition to the methods and properties below, it contains the same methods and properties as a `Node`.

Properties

Property	Type	Description
`material`	material	The `Mesh` object's default `Material`.
`renderMode`	string	The `Mesh` object's rendering style, which can be one of the following values: • `"default"` • `"bounding box"` • `"transparent bounding box"` • `"transparent bounding box outline"` • `"vertices"` • `"shaded vertices"` • `"wireframe"` • `"shaded wireframe"` • `"solid"` • `"transparent"` • `"solid wireframe"` • `"transparent wireframe"` • `"illustration"` • `"solid outline"` • `"shaded illustration"` • `"hidden wireframe"`

computeBoundingBox

Acrobat 7.0.7

Computes the bounds of the `Node` object.

Syntax

```
computeBoundingBox()
```

Returns

A `BoundingBox` object.

MouseEvent

An object that is passed as an argument to the `onEvent` method of the `MouseEventHandler` object.

Properties

Property	Type	Access	Description
`canvas`	Canvas	R	The `Canvas` in which the `MouseEvent` took place.
`canvasPixelHeight`	integer	R	The height, measured in pixels, of the `Canvas` in which the `MouseEvent` took place.
`canvasPixelWidth`	integer	R	The width, measured in pixels, of the `Canvas` in which the `MouseEvent` took place.
`ctrlKeyDown`	Boolean	R	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed.
`currentTool`	string	R	The name of the current tool.
`hits`	Array	R	A set of `HitInfo` objects ordered by distance from nearest to furthest.
`isDoubleClick`	Boolean	R	Determines whether a double-click event occurred
`isMouseDown`	Boolean	R	Determines whether the mouse button was pressed
`isMouseHit`	Boolean	R	Determines whether the target is under the mouse cursor.
`isMouseMove`	Boolean	R	Determines whether the mouse position changed.
`isMouseOut`	Boolean	R	Determines whether the mouse position moved off the target.
`isMouseOver`	Boolean	R	Determines whether the mouse position moved onto the target.
`isMouseUp`	Boolean	R	Determines whether the mouse button was released.
`leftButtonDown`	Boolean	R	Determines whether the left mouse button was pressed.
`mouseX`	integer	R	The x position of the mouse cursor in the `Canvas`.
`mouseY`	integer	R	The y position of the mouse cursor in the `Canvas`.

Property	Type	Access	Description
`rightButtonDown`	Boolean	R	Version 7.0.1 Determines whether the right mouse button was pressed.
`shiftKeyDown`	Boolean	R	Determines whether the Shift key was pressed.

MouseEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when a mouse event occurs. The handler may be customized to filter out certain event types. Event handlers are registered with the Runtime `addEventHandler` method.

Properties

Property	Type	Access	Description
`onMouseDoubleClick`	Boolean	R/W	When set to true, the handler is called back when a mouse button is clicked twice in rapid successionon the target object. If no target is specified, the handler calls back on any double-click.
`onMouseDown`	Boolean	R/W	When set to true, the handler is called back when a mouse button is initially pressed while the cursor is over the target object. If no target is specified, the handler calls back on any button press.
`onMouseHit`	Boolean	R/W	When set to true, the handler is called back continuously when the cursor is over the target object. In the case of onMouseHit, it does not matter if the target object is behind another object in the scene. The list of resultant hit objects are provided in the `MouseEvent hits` property.
`onMouseMove`	Boolean	R/W	When set to true, the handler is called back when the cursor moves over the target object. If no target is specified, the handler calls back on any mouse movement over the 3D annotation.
`onMouseOut`	Boolean	R/W	When set to true, the handler is called back once when the cursor moves off of the target object. To be called back, the target must be the frontmost object. To exclude objects, use the `Node hitEnabled` property.
`onMouseOver`	Boolean	R/W	When set to true, the handler is called back once when the cursor moves over the target object.
`onMouseUp`	Boolean	R/W	When set to true, the handler is called back when a mouse button is initially released. If a target is specified, it calls back only when the cursor is over the handler's target.

Property	Type	Access	Description
`reportAllTargets`	Boolean	R/W	Determines whether a hit test is performed. When set to `false`, a hit test is not performed except on a mouse-down or mouse-up event. This is an optimization feature because the current hit test is extremely expensive on complex models. When set to `false`, the following events are not reported because they depend on hit testing: • `mouse-hit` • `mouse-move` • `mouse-out` • `mouse-over`
`target`	object	R/W	The `Mesh` or `Background` object on which the mouse event occurs.

MouseEventHandler

A constructor that creates a new `MouseEventHandler` object.

Syntax

```
new MouseEventHandler()
```

Returns

A `MouseEventHandler` object.

onEvent

A method that is called when a mouse event occurs.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `MouseEvent` object representing the event.

Returns

undefined

Node

An object within the `Scene` hierarchy (a `SceneObject`) that has a 3D representation. The following objects are considered `Node` objects:

- `Bone`
- `Camera`
- `ClippingPlane`
- `Dummy`
- `Light`
- `Mesh`
- `Procedural`

To obtain a `Node` object's type, use the standard JavaScript `constructor` property. For example, the following syntax prints the `Node` object's type to the console:

```
console.println(myNode.constructor.name);
```

In addition to the methods and properties below, it contains the same methods and properties as a `SceneObject`.

Properties

Property	Type	Access	Description
`firstChild`	Node (if the first child exists), None otherwise	R	The `Node` object's first child.
`hitEnabled`	Boolean	R/W	Determines whether the `Node` is included in hit tests. The default value is `true`.
`info`	string	R	Acrobat 7.0.7 Information associated with the `Node` .
`metadataString`	string	R	Acrobat 8.1 A string containing `Node`-specific metadata.
`nextSibling`	Node (if the next sibling exists), None otherwise	R	The next sibling.
`opacity`	number	R/W	Acrobat 7.0.7 The `Node` opacity. A value from 0 to 1, where 1 is completely opaque.
`parent`	object	R	The `Node` object's parent `Node` or `Scene` .
`transform`	Matrix4x4	R	The local to world transformation matrix for the `Node` .

Property	Type	Access	Description
wireframeColor	Color	R	The Color object used to determine the wireframe appearance.
visible	Boolean	R/W	Determines whether the Node object should be shown. This property applies to mesh notes only. For example, modifying the empty parent node of a mesh tree has no effect on the child mesh tree items. In such cases it is recommended that you modify a parent node that is also a mesh node, and the child mesh items will have the same value for this property.

detachFromCurrentAnimation

Removes the ability of the currently active Animation of the Node object to transform the Node.

Syntax

```
detachFromCurrentAnimation()
```

Returns

undefined

Procedural

Deprecated

A `Node` object used to represent procedurally created geometry, such as constructive solid geometry (CSG) solids, procedural spheres, or NURB objects (a 3D curve or surface). A `Procedural` object contains the same methods and properties as a `Node`.

Quaternion

Represents a rotation in 3D space, and allows for smooth interpolation (blending) between orientations of subjects. A `Quaternion` is typically used for animating a `Camera` or `Mesh` over time, and can be converted to and from angles of rotation about the axes.

Quaternion

A constructor that initializes the object with the identity matrix.

Syntax

```
new Quaternion()
```

Returns

A `Quaternion` object.

Quaternion

A constructor that initializes the object with the specified rotation matrix.

Syntax

```
new Quaternion(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object representing the rotation matrix.

Returns

A `Quaternion` object.

Quaternion

A constructor that initializes the object with the specified `Quaternion`.

Syntax

```
new Quaternion(quaternion)
```

Parameters

`quaternion`	A `Quaternion` object used to initialize the new object.

Returns

A `Quaternion` object.

interpolate

Creates a `Quaternion` object interpolated from the current and specified `Quaternion` objects and `a`.

Syntax

```
interpolate(quaternion, a)
```

Parameters

`quaternion`	A `Quaternion` object used to interpolate the new object.
`a`	A number value, from `0.0` to `1.0`, that specifies the degree (percentage) of interpolation. A value of `0.5` represents an interpolation halfway between the current and specified `Quaternion` objects.

Returns

A `Quaternion` object.

interpolateInPlace

Creates a `Quaternion` object interpolated from the current and specified `Quaternion` objects and `a`, and updates the current `Quaternion` object with the new value.

Syntax

```
interpolateInPlace(quaternion, a)
```

Parameters

`quaternion`	A `Quaternion` object used to interpolate the new object.
`a`	A number value, from `0.0` to `1.0`, that specifies the degree (percentage) of interpolation. A value of `0.5` represents an interpolation halfway between the current and specified `Quaternion` objects.

Returns

A `Quaternion` object.

normalize

Normalizes the `Quaternion` object

Syntax

```
normalize()
```

Returns

undefined

RenderEvent

An object that is passed as an argument to the `onEvent` method of the `RenderEventHandler` object .

Properties

Property	Type	Access	Description
`canvas`	Canvas	R	The `Canvas` that is the target of the `RenderEvent` .
`canvasPixelHeight`	integer	R	The height, measured in pixels, of the `Canvas` for which the `RenderEvent` is intended.
`canvasPixelWidth`	integer	R	The width, measured in pixels, of the `Canvas` for which the `RenderEvent` is intended.
`currentTool`	string	R	The name of the current tool.

RenderEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the Runtime `addEventHandler` method. It issues a callback just before each `Canvas` is rendered.

RenderEventHandler

A constructor that creates a new `RenderEventHandler` object.

Syntax

```
new RenderEventHandler()
```

Returns

A `RenderEventHandler` object.

onEvent

A method that is called immediately before the `Canvas` is rendered.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `RenderEvent` object representing the event.

Returns

undefined

RenderOptions

An object that describes the style with which to render `Node` objects in the `Scene`.

Properties

Property	Type	Access	Description
`boundingBoxColor`	Color	R	A `Color` object to be applied to the bounding box.
`clippingPlaneColor`	Color	R	A `Color` object to be applied to the clipping plane.
`clippingPlaneIntersectionColor`	Color	R	A `Color` object to be applied to the clipping plane intersection.
`defaultAmbientColor`	Color	R	A `Color` object to be applied to the default ambient `Material`.
`defaultDiffuseColor`	Color	R	A `Color` object to be applied to the default diffuse `Material`.
`defaultEmissiveColor`	Color	R	A `Color` object to be applied to the default emissive `Material`.
`defaultSpecularColor`	Color	R	A `Color` object to be applied to the default specular `Material`.
`illustrationRenderModeFaceColor`	Color	R	Acrobat 7.0.7 The color of the faces when the render mode is Illustration.
`illustrationRenderModeLineColor`	Color	R	A `Color` object to be applied to the illustration lines.
`pointsRenderModeColor`	Color	R	A `Color` object to be applied to the vertices in point render mode.
`shadedIllustrationRenderModeLineColor`	Color	R	A `Color` object to be applied to the shaded illustration lines.

Property	Type	Access	Description
`solidGridColorEven`	Color	R	Acrobat 7.0.7 The color of the even squares of the checkered grid when drawn in solid mode.
`solidGridColorOdd`	Color	R	Acrobat 7.0.7 The color of the odd squares of the checkered grid when drawn in solid mode.
`solidRenderModeLineColor`	Color	R	A `Color` object to be applied to the solid or transparent lines in render mode.
`transparentBoundsRenderModeColor`	Color	R	A `Color` object to be applied to the transparent bounding box.
`transparentGridColorEven`	Color	R	Acrobat 7.0.7 The color of the even squares of the checkered grid when drawn in transparent mode.
`transparentGridColorOdd`	Color	R	Acrobat 7.0.7 The color of the odd squares of the checkered grid when drawn in transparent mode.
`wireframeRenderModeColor`	Color	R	Acrobat 7.0.7 The color of the wires when the render mode is Wireframe.
`xAxisColor`	Color	R	Acrobat 7.0.7 The color of the x axis.
`yAxisColor`	Color	R	Acrobat 7.0.7 The color of the y axis.
`zAxisColor`	Color	R	Acrobat 7.0.7 The color of the z axis.

Resource

An object that creates an abstraction for loading behavior in files and streams.

Properties

Property	Type	Access	Description
`type`	string	R	The type of `Resource` object, which can be one of the following values: • `"image"` • `"model"` • `"flash"` (Acrobat 9.0) • `"unknown"`
`TYPE_IMAGE`	string	R	Acrobat 7.0.7 A string constant for the `Resource` type of `"image"`.
`TYPE_MODEL`	string	R	Acrobat 7.0.7 A string constant for the `Resource` type of `"model"`.
`TYPE_UNKNOWN`	string	R	Acrobat 7.0.7 A string constant for the `Resource` type of `"unknown"`.
`TYPE_FLASH`	string	R	Acrobat 9.0 A string constant for the Resource type of `"flash"`.

Resource

A constructor that creates a new `Resource`.

Syntax

```
new Resource(pathname)
```

Parameters

`pathname`	A string representing the path of the file or stream. Can load embedded resources only from within the PDF file. The pathname string must start with pdf://.

Returns

A `Resource` object.

Runtime

An object that represents the run-time instance of the player. Each `Runtime` object can have its own unique script engine and set of annotations. The variable `runtime` is a global reference to this object.

Properties

Property	Type	Access	Description
`BUTTON_TYPE_PUSH`	string	R	Acrobat 7.0.7 A string constant for the custom tool button type of push button. It is used with the `addCustomToolButton` method.
`BUTTON_TYPE_TOOL`	string	R	Acrobat 7.0.7 A string constant for the custom button type of tool button. It is used with the `addCustomToolButton` method.
`canvasCount`	number	R	Acrobat 8.1 The number of Canvases that are attached to the active 3D annotation.
`ctrlKeyDown`	Boolean	R	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed.
`eventHandlerCount`	integer	R	The number of registered event handlers.
`instances`	Array	R	Acrobat 7.0.7 An array of JavaScript `Annot3D` objects that are attached to the 3D script context.
`MENU_ITEM_TYPE_CHECKED`	string	R	Acrobat 7.0.7 A string constant for the custom menu item type of checked. It is used with the `addCustomMenuItem` method.
`MENU_ITEM_TYPE_DEFAULT`	string	R	Acrobat 7.0.7 A string constant for the custom menu item type of default. It is used with the `addCustomMenuItem` method.
`overrideFlyTool`	Boolean	R/W	Acrobat 9.0 Determines whether to override the built-in Fly tool behavior.
`overrideNavTools`	Boolean	R/W	Determines whether to disable all default navigation behavior. **Note:** Setting this property does not prevent view changes.

Property	Type	Access	Description
`overridePanTool`	Boolean	R/W	Determines whether to override the built-in Pan tool behavior. **Note:** Setting this property does not affect the pan behavior of other navigation tools.
`overrideRotateTool`	Boolean	R/W	Determines whether to override the built-in Rotate tool behavior.
`overrideSelection`	Boolean	R/W	Acrobat 7.0.7 Determines whether to override the built-in Selection tool behavior.
`overrideSpinTool`	Boolean	R/W	Acrobat 8.0 Determines whether to override the built-in Spin tool behavior.
`overrideViewChange`	Boolean	R/W	Determines whether to override the setting of Views from Acrobat.
`overrideWalkTool`	Boolean	R/W	Determines whether to override the built-in Walk tool behavior.
`overrideScrollWheel`	Boolean	R/W	Acrobat 8.1 Determines whether to override the built-in scroll-wheel behavior.
`overrideZoomTool`	Boolean	R/W	Determines whether to override the built-in Zoom tool behavior. **Note:** Setting this property does not affect the zoom behavior of other navigation tools.
`scrollWheelSpeed`	number	R/W	Acrobat 8.1 A speed multiplier for the value of the scroll-wheel motion.
`shiftKeyDown`	Boolean	R	Determines whether the Shift key was pressed.

Property	Type	Access	Description
speedThreshold	number	R/W	Acrobat 8.1 A length (based upon the diagonal of the scene's bounding box) under which the Walk tool's motion is scaled relative to the size of the model. The Walk tool's motion is constant based upon the scene's scale factor, such that it emulates a natural pace relative to the model's size. This works well for architectural models that are created with a defined scale. However, the walk motion is too quick for very small models.
strafeSpeed	number	R/W	Acrobat 8.1 A speed multiplier for the lateral motion while using the Walk tool.
TOOL_NAME_FLY	string	R	Acrobat 9.0 A string constant for the name of the fly tool. Its value is "Fly".
TOOL_NAME_MEASURE	string	R	Acrobat 7.0.7 A string constant for the name of the measure tool. Its value is "Measure".
TOOL_NAME_PAN	string	R	Acrobat 7.0.7 A string constant for the name of the pan tool. Its value is "Pan".
TOOL_NAME_ROTATE	string	R	Acrobat 7.0.7 A string constant for the name of the rotate tool. Its value is "Rotate".
TOOL_NAME_SPIN	string	R	Acrobat 8.0 A string constant for the name of the Spin tool. Its value is "Spin".
TOOL_NAME_WALK	string	R	Acrobat 7.0.7 A string constant for the name of the walk tool. Its value is "Walk".
TOOL_NAME_ZOOM	string	R	Acrobat 7.0.7 A string constant for the name of the zoom tool. Its value is "Zoom".
version	number	R	The number corresponding to the version of the Runtime system.

Property	Type	Access	Description
viewCount	integer	R	Acrobat 9.0 The number of named views for the annotation.
walkSpeed	number	R/W	Acrobat 8.1 A speed multiplier for the forward/backward motion while using the Walk tool.

addCustomMenuItem

Creates a custom menu item in the 3D annotation context menu.

Syntax

```
addCustomMenuItem(name, label, type, checkedState)
```

Parameters

name	A string identifying the menu item.
label	A string appearing on the menu item.
type	A string indicating whether it is a checked menu item. A checked menu item has a check mark toggle next to it. Its possible values are: • "default" • "checked"
checkedState	A Boolean value indicating the state of a checked menu item.

Returns

undefined

addCustomToolButton

Creates a custom tool button in the 3D toolbar.

Syntax

```
addCustomToolButton(name, label, type)
```

Parameters

name	A string identifying the tool button.

`label`	A string appearing on the tool button.
`type`	A string indicating whether it is a tool button or a push button. Its possible values are: • `"tool button"` • `"push button"`

Returns

undefined

addEventHandler

Registers the provided event handler.

Syntax

```
addEventHandler(eventHandler)
```

Parameters

`eventHandler`	The event handler object to be registered.

Returns

undefined

disableTool

Disables the tool with the specified ID.

Syntax

```
disableTool(toolName)
```

Parameters

`toolName`	A string identifying the tool.

Returns

undefined

enableTool

Enables the tool with the specified ID.

Syntax

```
enableTool(toolName)
```

Parameters

`toolName`	A string identifying the tool.

Returns

undefined

getEventHandler

Obtains the event handler corresponding to the specified index.

Syntax

```
getEventHandler(index)
```

Parameters

`index`	An integer identifying the event handler.

Returns

An event handler object.

getRendererName

Obtains the name of the current renderer.

Syntax

```
getRendererName()
```

Returns

A string containing the current renderer's name.

getView

Acrobat 9.0

Gets the indicated view for the annotation by its index.

See the related method, `setView`, for setting the view by its index.

Syntax

```
getView( index )
```

Parameters

index	The integer index of the view.

Returns

View

getView

Acrobat 9.0

Gets the indicated view for the annotation by its name.

See the related method, setView, for setting the view by its name.

Syntax

```
getView( name )
```

Parameters

name	The string name of the view.

Returns

View

pause

Acrobat 9.0

Pauses the runtime. This is the same as selecting the Pause toolbar button or menu item.

Syntax

```
pause()
```

Returns

undefined

play

Acrobat 9.0

Resumes playback of the runtime. This is the same as selecting the Play toolbar button or menu item.

Syntax

```
play()
```

Returns

undefined

refresh

Version 7.0.1

Marks the render area dirty so that it is redrawn. This is useful when something changes in the scene but the annotation is in a Loaded and not Live state.

Syntax

```
refresh()
```

Returns

undefined

removeEventHandler

Unregisters the specified event handler.

Syntax

```
removeEventHandler(handler)
```

Parameters

`handler`	An event handler object representing the event handler.

Returns

undefined

removeCustomMenuItem

Removes the custom menu item with the specified ID.

Syntax

```
removeCustomMenuItem(menuName)
```

Parameters

menuName	A string identifying the custom menu item.

Returns

undefined

removeCustomToolButton

Removes the custom tool button with the specified ID.

Syntax

```
removeCustomToolButton(toolName)
```

Parameters

toolName	A string identifying the custom tool button.

Returns

undefined

setCurrentTool

Sets the current tool to the one with the specified ID.

Syntax

```
setCurrentTool(toolName)
```

Parameters

toolName	A string identifying the tool.

Returns

undefined

setCustomMenuItemChecked

Acrobat 7.0.7

Sets the checked state of the provided custom menu item.

Syntax

```
setCustomMenuItemChecked( menuItemName, checkedState )
```

Parameters

menuItemName	A string identifying the name of the custom menu item.
checkedState	A Boolean value determining whether the menu should be checked.

Returns

undefined

setView

Acrobat 9.0.

Sets the current view for the annotation.

See the related method, getView, for getting the view by its index.

Syntax

```
setView( index, animate)
```

Parameters

index	The integer index of the view to be set .
animate	(Optional) A Boolean value, when true, indicates that the view should be animated to when set.

Returns

undefined

setView

Acrobat 9.0

Sets the current view for the annotation.

See the related method, getView, for getting the view by its name.

Syntax

```
setView( name, animate)
```

Parameters

`menuItemName`	The string name of the view to be set.
`checkedState`	(Optional) A Boolean value, when `true`, indicates that the view should be animated to when set.

Returns

undefined

Scene

An object that represents the hierarchy of the 3D related content, including `Animation`, `Light`, `Material`, and `Mesh` objects. The variable `scene` is a global reference to this object.

Related objects are `Animation`, `Light`, `Material` and `Mesh`.

Properties

Property	Type	Access	Description
`ambientIlluminationColor`	Color	R	Modulates the ambient `Color` of all materials.
`animations`	SceneObjectList	R	A list of all `Animation` objects.
`cameras`	SceneObjectList	R	A list of all `Camera` objects in the `Scene`.
`defaultRenderOptions`	RenderOptions	R	A set of all default rendering options for the `Scene`.
`gridMode`	string	R/W	Acrobat 7.0.7 The display style of the grid that represents a portion of the ground plane in the `Scene`. It can have one of the following values: • `"off"` (no grid) • `"wire"`(a wireframe grid) • `"solid"`(a solid checkerboard grid) • `"transparent"` (a semi-transparent checkerboard grid)
`GRID_MODE_OFF`	string	R	Acrobat 7.0.7 A string constant for the grid mode of `"off"`.
`GRID_MODE_SOLID`	string	R	Acrobat 7.0.7 A string constant for the grid mode of `"solid"`.
`GRID_MODE_TRANSPARENT`	string	R	Acrobat 7.0.7 A string constant for the grid mode of `"transparent"`.

Property	Type	Access	Description
GRID_MODE_WIRE	string	R	Acrobat 7.0.7 A string constant for the grid mode of "wire".
gridSize	number	R	Acrobat 7.0.7 The number of squares on the ground plane grid.
lengthUnits	number	R	The scale of a unit of length, specified in meters.
LIGHT_MODE_FILE	string	R	Acrobat 7.0.7 A string constant for the light mode of "file".
LIGHT_MODE_NONE	string	R	Acrobat 7.0.7 A string constant for the light mode of "none".
LIGHT_MODE_WHITE	string	R	Acrobat 7.0.7 A string constant for the light mode of "white".
LIGHT_MODE_DAY	string	R	Acrobat 7.0.7 A string constant for the light mode of "day".
LIGHT_MODE_BRIGHT	string	R	Acrobat 7.0.7 A string constant for the light mode of "bright".
LIGHT_MODE_RGB	string	R	Acrobat 7.0.7 A string constant for the light mode of "rgb".
LIGHT_MODE_NIGHT	string	R	Acrobat 7.0.7 A string constant for the light mode of "night".
LIGHT_MODE_BLUE	string	R	Acrobat 7.0.7 A string constant for the light mode of "blue".
LIGHT_MODE_RED	string	R	Acrobat 7.0.7 A string constant for the light mode of "red".

Property	Type	Access	Description
LIGHT_MODE_CUBE	string	R	Acrobat 7.0.7 A string constant for the light mode of "cube".
LIGHT_MODE_CAD	string	R	Acrobat 7.0.7 A string constant for the light mode of "cad".
LIGHT_MODE_HEADLAMP	string	R	Acrobat 7.0.7 A string constant for the light mode of "headlamp".
lights	SceneObjectList	R	A list of all Light objects in the Scene.
lightScaleFactor	number	R/W	A uniform scale factor for all Light objects in the Scene.
lightScheme	string	R/W	Acrobat 7.0.7 The current, preconfigured lighting scheme for the Scene. It can take one of the following values: • "file" • "none" • "white" • "day" • "bright" • "rgb" • "night" • "blue" • "red" • "cube" • "cad" • "headlamp"
materials	SceneObjectList	R	A list of all Material objects.
meshes	SceneObjectList	R	A list of all Mesh objects in the Scene.

Property	Type	Access	Description
`nodes`	SceneObjectList	R	A list of all `Node` objects except the default `Camera` and default `Light` objects.
`outlineAngle`	number	R/W	Acrobat 7.0.7 The crease angle (in degrees) for the appearance of lines in Illustration render modes.
`showGrid`	Boolean	R/W	Acrobat 7.0.7 Determines whether the ground plane grid is displayed.
`renderDoubleSided`	Boolean	R/W	Acrobat 8.1 Toggles if backfacing polygons are rendered.
`renderMode`	string	R/W	Acrobat 7.0.7 The default rendering type for all objects in the `Scene`, which can be one of the following values: • `"default"` • `"bounding box"` • `"transparent bounding box"` • `"transparent bounding box outline"` • `"vertices"` • `"shaded vertices"` • `"wireframe"` • `"shaded wireframe"` • `"solid"` • `"transparent"` • `"solid wireframe"` • `"transparent wireframe"` • `"illustration"` • `"solid outline"` • `"shaded illustration"` • `"hidden wireframe"`

Property	Type	Access	Description
RENDER_MODE_DEFAULT	string	R	Acrobat 7.0.7 A string constant for the render mode of "default".
RENDER_MODE_BOUNDING_BOX	string	R	Acrobat 7.0.7 A string constant for the render mode of "bounding box".
RENDER_MODE_TRANSPARENT_BOUNDING_BOX	string	R	Acrobat 7.0.7 A string constant for the render mode of "transparent bounding box".
RENDER_MODE_TRANSPARENT_BOUNDING_BOX_OUTLINE	string	R	Acrobat 7.0.7 A string constant for the render mode of "transparent bounding box outline".
RENDER_MODE_VERTICES	string	R	Acrobat 7.0.7 A string constant for the render mode of "vertices".
RENDER_MODE_SHADED_VERTICES	string	R	Acrobat 7.0.7 A string constant for the render mode of "shaded vertices".
RENDER_MODE_WIREFRAME	string	R	Acrobat 7.0.7 A string constant for the render mode of "wireframe".
RENDER_MODE_SHADED_WIREFRAME	string	R	Acrobat 7.0.7 A string constant for the render mode of "shaded wireframe".
RENDER_MODE_SOLID	string	R	Acrobat 7.0.7 A string constant for the render mode of "solid".

Property	Type	Access	Description
RENDER_MODE_TRANSPARENT	string	R	Acrobat 7.0.7 A string constant for the render mode of "transparent".
RENDER_MODE_SOLID_WIREFRAME	string	R	Acrobat 7.0.7 A string constant for the render mode of "solid wireframe".
RENDER_MODE_TRANSPARENT_WIREFRAME	string	R	Acrobat 7.0.7 A string constant for the render mode of "transparent wireframe".
RENDER_MODE_ILLUSTRATION	string	R	Acrobat 7.0.7 A string constant for the render mode of "illustration".
RENDER_MODE_SOLID_OUTLINE	string	R	Acrobat 7.0.7 A string constant for the render mode of "solid outline".
RENDER_MODE_SHADED_ILLUSTRATION	string	R	Acrobat 7.0.7 A string constant for the render mode of "shaded illustration".
RENDER_MODE_HIDDEN_WIREFRAME	string	R	Acrobat 7.0.7 A string constant for the render mode of "hidden wireframe".
selectedNode	Node	R/W	Acrobat 8.1 The currently selected Node.
showAxes	Boolean	R/W	Acrobat 7.0.7 Determines whether the world axes are displayed.
showOrientationAxes	Boolean	R/W	Acrobat 9.0 Determines whether the orientation axes are displayed.

Property	Type	Access	Description
smoothing	Boolean	R/W	Acrobat 7.0.7 When true, smoothing is enabled for meshes in the scene.
smoothingAngle	number	R/W	Acrobat 7.0.7 The default smoothing angle (in degrees) for meshes in the scene.
smoothingOverride	Boolean	R/W	Acrobat 7.0.7 When set to true, overrides the smoothing values from the loaded model file.

activateAnimation

Sets the given Animation to be active on its Node objects.

Syntax

```
activateAnimation(animation)
```

Parameters

animation	The Animation object to be activated.

Returns

undefined

addFlashForeground

Acrobat 9.0

Adds the provided FlashMovie as a foreground element within the 3D scene.

Syntax

```
addFlashForeground(movie)
```

Parameters

movie	The FlashMovie to be added as a foreground element.

Returns

undefined

addModel

Adds a model `Resource` to the top level of the `Scene`.

Syntax

```
addModel(modelRes)
```

Parameters

`modelRes`	The `Resource` object to be added.

Returns

A `Node` object representing the top-level `Mesh` of the loaded model.

createClippingPlane

Creates a new clipping plane.

Syntax

```
createClippingPlane()
```

Returns

A `ClippingPlane` object.

createLight

Creates a new `Light` and attaches it to the `Scene` .

Syntax

```
createLight()
```

Returns

A `Light` object.

createSquareMesh

Creates a `Mesh` that is a unit square. The default UV parameterization fits once in U and V.

Syntax

```
createSquareMesh(sizeX, sizeY, name)
```

Parameters

`sizeX`	Model units in the x direction used to size the `Mesh`.
`sizeY`	Model units in the y direction used to size the `Mesh`.
`name`	(Optional) The name that is assigned to the `Mesh`.

Returns

A `Mesh` object.

computeBoundingBox

Computes the `BoundingBox` of the `Scene` .

Syntax

```
computeBoundingBox()
```

Returns

A `BoundingBox` object.

update

Applies all changes to the `Scene`.

Syntax

```
update()
```

Returns

undefined

SceneObject

The base type for objects within the `Scene`, including `Animation`, `Material`, and `Node` objects.

Related objects are `Scene`, `Animation`, `Light`, `Material`, and `Mesh`.

Properties

Property	Type	Description
`name`	string	The name of the `SceneObject` object.
`objectGUID`	string	Deprecated A value that uniquely identifies the `SceneObject` in custom data. This property has a default value.
`objectID`	integer	An unsigned 32-bit value that uniquely identifies the `SceneObject`. This property can be assigned, but it does not have a default value. It always returns `0`.

SceneObjectList

A structure that contains references to several `SceneObject` objects.

Properties

Property	Type	Access	Description
`count`	integer	R	The number of elements in the `SceneObjectList`.

getByGUID

Deprecated

Obtains the specified `SceneObject` object from the list.

Syntax

```
getByGUID(guid)
```

Parameters

`guid`	A string representing the GUID for the specified element.

Returns

A `SceneObject` object.

getByID

Obtains the specified `SceneObject` object from the list

Syntax

```
getByID(id)
```

Parameters

`id`	An integer representing the object identifier for the specified `SceneObject` object.

Returns

A `SceneObject` object.

getByIndex

Obtains the specified `SceneObject` object from the list.

Syntax

```
getByIndex(index)
```

Parameters

index	An integer representing the index of the specified SceneObject object.

Returns

A SceneObject object.

getByName

Obtains the specified SceneObject object from the list.

Syntax

```
getByName(name)
```

Parameters

name	A string representing the name of the specified SceneObject object.

Returns

A SceneObject object.

removeAll

Deprecated

Removes all the SceneObject objects from the list, but does not delete them from the Scene.

Syntax

```
removeAll()
```

Returns

undefined

removeByIndex

Deprecated

Removes the specified SceneObject object from the list, but does not delete it from the Scene.

Syntax

```
removeByIndex(index)
```

Parameters

index	An index to the specified element.

Returns

undefined

removeItem

Deprecated

Removes a SceneObject object from the list, but does not delete it from the Scene.

Syntax

```
removeItem(scene)
```

Parameters

scene	A scene object that is to be removed.

Returns

undefined

ScrollWheelEvent

(Acrobat 8.1) An object that is passed as an argument to the `onEvent` method of the `ScrollWheelEventHandler` object. A `ScrollWheelEvent` object is created when the mouse scroll wheel is activated over an active 3D `Canvas` object.

Properties

Property	Type	Access	Description
`canvas`	Canvas	R	The `Canvas` in which the `ScrollWheelEvent` took place.
`canvasPixelHeight`	integer	R	The height, measured in pixels, of the `Canvas` in which the `ScrollWheelEvent` took place.
`canvasPixelWidth`	integer	R	The width, measured in pixels, of the `Canvas` in which the `ScrollWheelEvent` took place.
`ctrlKeyDown`	Boolean	R	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed.
`currentTool`	string	R	The name of the current tool.
`deltaY`	number	R	The amount the scroll wheel was moved in the Y direction.
`shiftKeyDown`	Boolean	R	Determines whether the Shift key was pressed.

ScrollWheelEventHandler

(Acrobat 8.1) An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime` method `addEventHandler`.

ScrollWheelEventHandler

A constructor that creates a new `ScrollWheelEventHandler`.

Syntax

```
new ScrollWheelEventHandler()
```

Returns

A `ScrollWheelEventHandler` object.

onEvent

A method that is called when the scroll wheel is used in an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `ScrollWheelEvent` object representing the event.

Returns

undefined

SelectionEvent

(Acrobat 8.1) An object that is passed as an argument to the `onEvent` method of the `SelectionEventHandler` object.

A `SelectionEvent` object is created when an object is selected from an active 3D `Canvas` object or from a model tree. If the selection is made from a `Canvas` object, a `MouseEvent` is also created.

Properties

Property	Type	Access	Description
`node`	Node	R	The Node that is the target of the selection change.
`selected`	Boolean	R	The selected state of the target Node.

SelectionEventHandler

(Acrobat 8.1) An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime` method `addEventHandler`.

SelectionEventHandler

A constructor that creates a new `SelectionEventHandler` object.

Syntax

```
new SelectionEventHandler()
```

Returns

A `SelectionEventHandler` object.

onEvent

A method that is called when the selection state changes in an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `ScrollWheelEvent` object representing the event.

Returns

undefined

StateEvent

Acrobat 9.0

An object that is passed as an argument to the `onEvent` method of the `StateEventHandler` object. A `StateEvent` object is created when state data must be stored or loaded for the scene, such as when a new comment view is created or invoked for the annotation.

Properties

Property	Type	Access	Description
`stateString`	string	R	If the `SaveEvent type` is `"load"`, this property contains the state data that was stored as part of the corresponding `"save" StateEvent`. If the `SaveEvent type` is `"save"`, the `stateString` is undefined.
`type`	string	R	The type of `StateEvent`, this property has a value of either `"load"` or `"save"`.
`TYPE_LOAD`	string	R	A string constant for the `StateEvent` type of `"load"`. The state data that was stored as part of the original `stateEvent`.
`TYPE_SAVE`	string	R	A string constant for the `StateEvent` type of `"save"`.

StateEventHandler

Acrobat 9.0

An object that exposes a callback mechanism that allows a function to be evaluated when a state event occurs. Event handlers are registered with the `Runtime` method `addEventHandler`.

onEvent

A method that is called when state data must be stored or loaded for the annotation. The return value is stored as the `stateString` for the given `StateEvent`.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `StateEvent` object representing the event.

Returns

string or undefined

StateEventHandler

The constructor that creates a new `StateEventHandler`.

Syntax

```
new StateEventHandler()
```

Returns

A `StateEventHandler` object.

Texture

A `Texture` object represents the mapping of a texture. All `Texture` properties have read-write permissions.

Properties

Property	Type	Description
`amount`	number	The degree to which the `Texture` is applied, which can be a value from `0.0` to `1.0`.
`angle`	number	The rotation angle, measured in degrees, of the map.
`clampU`	Boolean	Determines whether the map should be clamped in the U direction.
`clampV`	Boolean	Determines whether the map should be clamped in the V direction.
`image`	Image	Acrobat 7.0.7 The `Image` to be used by the `Texture`.
`modulate`	Boolean	Determines whether to set the blend mode of the `Texture` with its corresponding color.
`uOffset`	number	The U-offset of the given map.
`uScale`	number	The U-scale of the given map.
`use3DSStyleMapping`	Boolean	Determines whether to use 3D Studio style map parameterizations.
`vOffset`	number	The V-offset of the given map.
`vScale`	number	The V-scale of the given map.

getImage

Deprecated

Gets the `Image` currently used by the `Texture`.

Syntax

```
getImage()
```

Returns

The Image currently being used.

setImage

Deprecated

Sets the `Image` to be used by the `Texture`.

Syntax

```
setImage(image)
```

Parameters

`image`	The `Image` to be used.

Returns

undefined

TimeEvent

An object that is passed as an argument to the `TimeEventHandler` object's `onEvent` method.

Properties

Property	Type	Access	Description
`deltaTime`	number	R	The difference between the current time and the last time.
`time`	number	R	The current time.

TimeEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime addEventHandler` method.

TimeEventHandler

A constructor that creates a new `TimeEventHandler` object.

Syntax

```
new TimeEventHandler()
```

Returns

A `TimeEventHandler` object.

onEvent

A method that is called when simulation time is incremented in an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `TimeEvent` object representing the event.

Returns

undefined

ToolEvent

An object that is passed as an argument to the `onEvent` method of the `ToolEventHandler` object.

Properties

Property	Type	Access	Description
`canvas`	Canvas	R	The `Canvas` that is the target of the `ToolEvent`.
`canvasPixelHeight`	integer	R	The height, measured in pixels, of the `Canvas` for which the `ToolEvent` is intended.
`canvasPixelWidth`	integer	R	The width, measured in pixels, of the `Canvas` for which the `ToolEvent` is intended.
`currentTool`	string	R	The name of the current tool.
`toolName`	string	R	The name of the tool that was clicked.

ToolEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime` `addEventHandler` method.

ToolEventHandler

A constructor that creates a new `ToolEventHandler` object.

Syntax

```
new ToolEventHandler()
```

Returns

A `ToolEventHandler` object.

onEvent

A method that is called when a tool button is pressed on the 3D toolbar.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `ToolEvent` object representing the event.

Returns

undefined

Vector3

An object comprised of three values that represent a point in space or a direction and magnitude.

Properties

Property	Type	Access	Description
`x`	number	R/W	The x component of the `Vector3` object.
`y`	number	R/W	The y component of the `Vector3` object.
`z`	number	R/W	The z component of the `Vector3` object.
`length`	number	R	The length of the `Vector3` object.

Vector3

A constructor that initializes the new object to `(0.0, 0.0, 0.0)`.

Syntax

```
new Vector3()
```

Returns

A `Vector3` object.

Vector3

A constructor that initializes the new object to the specified components.

Syntax

```
new Vector3(x, y, z)
```

Parameters

`x`	The x component used to initialize the new object.
`y`	The y component used to initialize the new object.
`z`	The z component used to initialize the new object.

Returns

A `Vector3` object.

add

Adds the specified `Vector3` to the current one.

Syntax

```
add(offset)
```

Parameters

`offset`	The `Vector3` object to be added to the current one.

Returns

A `Vector3` object.

addInPlace

Adds the specified `Vector3` to the current one, and updates the current `Vector3` with the resulting value.

Syntax

```
addInPlace(offset)
```

Parameters

`offset`	The `Vector3` object to be added to the current one.

Returns

undefined

addScaled

Adds the specified `Vector3` with the scaled offset to the current one.

Syntax

```
addScaled(offset, scale)
```

Parameters

`offset`	The `Vector3` object to be added to the current one.
`scale`	The scaling factor for the `offset`.

Returns

A `Vector3` object.

addScaledInPlace

Adds the specified `Vector3` with the scaled offset to the current one, and updates the current `Vector3` with the resulting value.

Syntax

```
addScaledInPlace(offset, scale)
```

Parameters

`offset`	The `Vector3` object to be added to the current one.
`scale`	The scaling factor for the `offset`.

Returns

undefined

blend

Blends the current and specified `Vector3` by the specified degree.

Syntax

```
blend(vec, blendFactor)
```

Parameters

`vec`	The `Vector3` object to be blended with the current one.
`blendFactor`	The degree of blending to be applied, which can be a value from `0.0` to `1.0`.

Returns

A `Vector3` object.

blendInPlace

Blends the current and specified `Vector3` by the specified degree, and updates the current `Vector3` with the resulting value.

Syntax

```
blendInPlace(vec, blendFactor)
```

Parameters

`vec`	The `Vector3` object to be blended with the current one.
`blendFactor`	The degree of blending to be applied, which can be a value from `0.0` to `1.0`.

Returns

undefined

cross

Calculates the cross product between the specified `Vector3` and the current one.

Syntax

```
cross(vec)
```

Parameters

`vec`	The `Vector3` object to be used in calculating the cross product.

Returns

A `Vector3` object.

dot

Calculates the dot product between the specified `Vector3` and the current one.

Syntax

```
dot(vec)
```

Parameters

`vec`	The `Vector3` object to be used in calculating the dot product.

Returns

A number value representing the scalar dot product.

normalize

Normalizes the current `Vector3`.

Syntax

```
normalize()
```

Returns

undefined

scale

Scales the current `Vector3`.

Syntax

```
scale(scale)
```

Parameters

`scale`	A number value used to scale the current `Vector3`.

Returns

A `Vector3` object.

scaleInPlace

Scales the current `Vector3`, and updates the current `Vector3` with the resulting value.

Syntax

```
scaleInPlace(scale)
```

Parameters

`scale`	A number value used to scale the current `Vector3`.

Returns

undefined

set

Sets the current `Vector3` to the value contained in the specified `Vector3`.

Syntax

```
set(vec)
```

Parameters

vec	The Vector3 used to set the current Vector3.

Returns

undefined

set

Acrobat 7.0.7

Sets the current Vector3 to the values contained in the specified components.

Syntax

```
set(x, y, z)
```

Parameters

x	The x component used to set the current Vector3.
y	The y component used to set the current Vector3.
z	The z component used to set the current Vector3.

Returns

undefined

set3

Deprecated

Sets the current Vector3 to the values contained in the specified components.

Syntax

```
set3(x, y, z)
```

Parameters

x	The x component used to set the current Vector3.
y	The y component used to set the current Vector3.
z	The z component used to set the current Vector3.

Returns

undefined

subtract

Subtracts the specified `Vector3` from the current one.

Syntax

```
subtract(offset)
```

Parameters

`offset`	The `Vector3` object to be subtracted from the current one.

Returns

A `Vector3` object.

subtractInPlace

Subtracts the specified `Vector3` from the current one, and updates the current `Vector3` with the resulting value.

Syntax

```
subtractInPlace(offset)
```

Parameters

`offset`	The `Vector3` object to be subtracted from the current one.

Returns

undefined

View

Acrobat 9.0

An object that represents a named view for the annotation.

See the `viewCount` property and the `getView` and `setView` methods of the `Runtime` object.

Properties

Property	Type	Access	Description
`name`	string	R	The name of the view.

3 New Features and Changes

This chapter summarizes the new features and changes introduced in Acrobat.

Acrobat X changes

There are no changes to JavaScript for Acrobat 3D Annotations in Acrobat X.

Acrobat 9.0 changes

This section describes the changes introduced in Acrobat 9 Pro Extended.

New objects

The following new objects are introduced to support Rich Media annotations and Geospatial features in Acrobat 9.0:

- FlashEvent object: This new object has these properties: command, target, type, TYPE_COMMAND, TYPE_PROGRESS, TYPE_STATECHANGE, and value.
- FlashEventHandler object: This new object has one property, target, and these methods: onEvent and FlashEventHandler.
- FlashMovie object: This new object has these properties: alignMode, ALIGN_MODE_BOTTOM, ALIGN_MODE_LEFT, ALIGN_MODE_RIGHT, ALIGN_MODE_TOP, backgroundColor, desiredResolutionX, desiredResolutionY, frameNum, hitEnabled, id, loop, opacity, percentLoaded, playing, quality, readyState, resolutionType, RESOLUTION_TYPE_CUSTOM, RESOLUTION_TYPE_MOVIE, RESOLUTION_TYPE_WINDOW, scaleMode, SCALE_MODE_EXACT_FIT, SCALE_MODE_NO_BORDER, SCALE_MODE_SHOW_ALL, totalFrames, x, and y. It also has these methods: FlashMovie, call, getVariable, gotoFrame, isPlaying, pan, play, rewind, setVariable, setZoomRect, stop, and zoom.
- StateEvent object: This new object has these properties: stateString, type, TYPE_LOAD, and TYPE_SAVE.
- StateEventHandler object: This new object has these methods: onEvent and StateEventHandler.
- View object: This new object has the name property.

Additional properties and methods in existing objects

- The Background object has one additional property, FlashMovie.
- The Resource object has additional properties: type (a new value of "flash") and TYPE_FLASH.
- This Material object has one additional method, attachFlashMovie.
- The Runtime object has additional properties: overrideFlyTool, TOOL_NAME_FLY, and viewCount. It also has additional methods: pause, play, getView, and setView.
- The Scene object has a new property, showOrientationAxes, and a new method, addFlashForeground.

APIs for versions earlier than 9.0

The following properties and methods were available but undocumented in earlier versions of this reference.

- Two properties were defined for the Camera object in version 7, absoluteBindingScale and useAbsoluteBinding.
- Two set methods were defined for the Matrix4x4 object in version 8.1, one takes an array argument and the other a list of numbers.
- The setCustomMenuItemChecked method was defined for the Runtime object in version 7.0.7.
- The opacity property was defined for the Node object in version 7.0.7.

API changes

- The computeBoundingBox method is documented as a method of the Mesh object, not the Node object, as previously published.
- The property previously documented for the Runtime object as overrideWheelSpeed was incorrect. It is now properly identified as overrideScrollWheel, and the description changed.
- The RENDER_MODE_SOLID_WIREFRAME property for the Scene object was incorrectly documented as RENDER_MODE_SHADED_SOLID_WIREFRAME.
- Deprecate the methods removeAll, removeByIndex, and removeItem for the SceneObjectList object. Their descriptions were modified to indicate that these methods remove elements from the SceneObjectList, but not from the Scene.

Acrobat 8.1 changes

This section describes the changes introduced in Acrobat 8.1.

New objects

The following objects are new: ScrollWheelEvent, ScrollWheelEventHandler, SelectionEvent, and SelectionEventHandler.

Additional properties in existing objects

The HitInfo object has additional properties: material, surfaceNormal, and textureCoordinate.

The Node object has an additional property: metadataString.

The Light object has an additional property: directionLocal (Acrobat 7, but previously undocumented).

The Runtime object has additional properties: canvasCount, overrideSpinTool, scrollWheelSpeed, speedThreshold, strafeSpeed, and walkSpeed.

The Scene object has additional properties: node and selected.

Deprecated objects or properties

The following APIs were deprecated:

`CamaraEvent.isNewCanvas` (a property)

`Dummy` (an object)

`Procedural` (an object)

`SceneObject.objectGUID` (a property)

`SceneObject.getByGUID` (a method)

Acrobat 8.0 changes

This section describes the changes introduced in Acrobat 8.0.

Additional properties in existing objects

The `Runtime` object has additional properties: `overrideSpinTool` and `TOOL_NAME_SPIN`.

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z