Expression and Scripting Improvements in the October 2017 release of After Effects CC

Introduction

This article documents the new expression and scripting functionality included in in the October 2017 release of After Effects CC (version 15.0).

Expression access to data in JSON files

After Effects can now import data files in JSON (JavaScript Object Notation) format. Data in a JSON file can be referenced via expressions to driv e animation in a composition.

Using JSON files (.json)

Import JSON files into your project as footage, the same way you would import a video, audio, or still image file. Like other footage, when a change is saved into a JSON file, After Effects will update the project to reflect that change.

You do not need to add JSON files to a composition, as they are referenced via expressions as footage objects, not layer objects. However, it is good general practice to add JSON files to any composition that references them, and is required if you export the composition as a Motion Graphics template or add the composition to the Adobe Media Encoder queue. Also, JSON files not added to a composition will be removed from the project when you choose Remove Unused Footage, Reduce Project, or Collect Files with the Collect Source Files option set to any option other than All.

To reference data in a JSON file, apply an expression to a property in your composition that you want the data to drive:

1. Use the `sourceData` footage attribute to return the data in a JSON file. For example:
   `var myData = footage("sample.json").sourceData;`
2. Reference the specific property inside the JSON data to return its value. For example, if your data contains latitude and longitude values, the values might be `myData.latitude` and `myData.longitude`. If your data came from motion capture hardware, the right foot's rotation value might be in a hierarchical structure like this: `myData.hips.RightUpLeg.RightLeg.RightFoot.ROTATION`

Another method of retrieving data from a JSON file is to use the `sourceText` footage attribute, which returns the contents of a JSON file as a text string. You can, for example, apply `sourceText` to a text layer's Source Text property to display the contents of a JSON file, like this: `foota` `ge("sample.json").sourceText`. Also, although less efficient than the `sourceData` method, you can use the JavaScript `eval()` function to convert the string into a data object, for example: `eval("var myData =" + footage("sample.json").sourceText);`

If you want to use data that is in a different format, like CSV, there are many conversion tools available to convert other data formats to JSON.

About the motion graphics JSON schema (.mgJSON)

Motion graphics JSON (MGJSON) is a new schema for the JSON format created by Adobe for the purpose of optimizing large sets of temporal data. Temporal data is data that changes over time, such as data recorded during motion tracking, GPS capture, or other device telemetry. The MGJSON schema defines structure for dynamic data streams so that After Effects can sample temporal data at a specific time value instead of sampling the entire data set. This improves performance of reading large data sets compared to other JSON structures.

Many of the expression methods for referencing data are specific to the MGJSON data structure, and will not work with other JSON files. These expression methods reference MGJSON data via footage objects. However, you don't necessarily need to write expressions that reference the footage object in order to use MGJSON data. Unlike other JSON files, there is a direct benefit to adding MGJSON files to a composition: After Effects represents the data structure of MGJSON files as layer properties in compositions. The data group properties can be referenced by expressions the same way you would reference a layer's transform group properties, masks, or effects. You can easily build complex animation based on MGJSON data by letting After Effects write expression code when you link to the data group properties using the expression pickwhip or the Edit > Copy with Property Links command.

Each individual data property in an MGJSON layer's data group has an expression applied which links that property to the source data in the MGJSON file. If the source MGJSON file is changed, the data property values will automatically update. There is also a new Create Keyframes From Data keyframe assistant, in the Animation > Keyframe Assistant menu, which freezes data property values as keyframes. The resulting keyframes are not linked to the source data. If the MGJSON file is changed and you want the keyframes to also change, use this command again to update the keyframes. Use this command instead of Convert Expressions to Keyframes, which will create keyframes on every frame instead of only where there are data points.

As of the writing of this article, the MGJSON schema has not yet been published. More information will be made available when the MGJSON schema is published.

Expression reference

Footage sourceText attribute

`{footageItem}``.sourceText`

Description

Returns the contents of a JSON file as a string.

The `eval()` method can be used to convert the string to aan array of sourceData objects, identical to the results of the `sourceData` attribute, from which the individual data streams can be referenced as hierarchal attributes of the data. For example:

```
var myData = eval(footage("sample.json").sourceText);
myData.sampleValue;
```

Type

String, the contents of the JSON file; read-only.

Footage sourceData attribute

`{footageItem}``.sourceData`

Description

Returns the data of a JSON file as an array of sourceData objects.

The structure of the JSON file will determine the size and complexity of the array.

Individual data streams can be referenced as hierarchal attributes of the data. For example, given a data stream named "Color", the following will return the value of Color from the first data object:

```
footage("sample.json").sourceData[0].Color
```

Typical use is to assign a JSON file's sourceData to a variable, and then reference the desired data stream. For example:

```
var myData = footage("sample.json").sourceData;
myData[0].Color;
```

Type

An array of sourceData objects; read-only.

Footage dataValue() method

`{footageItem}.dataValue(dataPath)`

Description

Returns the value of specificed static or dynamic data stream in a MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired data stream. For example:

- `footage("sample.mgjson").dataValue([0])` returns data of the first child
- `footage("sample.mgjson").dataValue([1][0])` returns data of the first child in the second group

Parameters

dataPath	Array, required. The path in the hierarchy to a static or dynamic data stream.

Returns

The value of the data stream.

Footage dataKeyCount() method

`{footageItem}.dataKeyCount(dataPath)`

Description

Returns the number of samples in a specificed dynamic data stream in a MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired dynamic data stream. For example:

- `footage("sample.mgjson").dataKeyCount([0])` returns the count of samples for the first child
- `footage("sample.mgjson").dataKeyCount([1][0])` returns the count of samples for the second group

Parameters

dataPath	Array, required. The path in the hierarchy to a dynamic data stream.

Returns

The number of samples in the dynamic data stream.

Footage dataKeyTimes() method

`{footageItem}.dataKeyTimes(dataPath, t0 = startTime, t1=endTime)`

Description

Returns the time in seconds for the samples of a specificed dynamic data stream in a MGJSON file.

Optionally specify the time span from which to return samples. By default the time for all samples between startTime and endTime in the dynamic data stream are returned, as defined by the data stream's samplesTemporalExtent property in the MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired dynamic data stream. The following example returns the times of samples between 1 second and 3 seconds for the first child:

```
footage("sample.mgjson").dataKeyTimes([0], 1, 3)
```

Parameters

dataPath	Array, required. The path in the hierarchy to a dynamic data stream.
t0	Number, optional. The start time, in seconds, of the span from which to return samples. Defaults to startTime.
t1	Number, optional. The end time, in seconds, of the span from which to return samples. Defaults to endTime.

Returns

Array of numbers representing the sample times.

Footage dataKeyValues() method

`{footageItem}.dataKeyValues(dataPath, t0 = startTime, t1=endTime)`

Description

Returns the values for the samples of a specificed dynamic data stream in a MGJSON file.

Optionally specify the time span from which to return samples. By default the time for all samples between startTime and endTime in the dynamic data stream are returned, as defined by the data stream's samplesTemporalExtent property in the MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired dynamic data stream. For example:

- `footage("sample.mgjson").dataKeyTimes([0], 1, 3)` returns the values of samples between 1 second and 3 seconds for the first child

Parameters

dataPath	Array, required. The path in the hierarchy to a dynamic data stream.
t0	Number, optional. The start time, in seconds, of the span from which to return samples. Defaults to startTime.
t1	Number, optional. The end time, in seconds, of the span from which to return samples. Defaults to endTime.

Returns

Array of numbers representing the sample values.

Building a custom expression function library in a JSON file

Because JSON files can contain JavaScript code, you can build a library of custom expression functions library into a JSON file. This can save you time if you write the same expression functions frequently. Instead of re-writing or copying and pasting a function's code into different expressions, you can write the function into a JSON file. Then the expressions only need to include code that references the function in the JSON file. Another benefit is that if you make changes to a function in the JSON file, all instances of the function will automatically update.

To automate making your custom expression function library available in new projects, import your JSON file into a template project and set that template project to load when you create a new project.

Here is an example of a JSON file named MyFunctions.json that contains a two functions for calculating the circumference of a circle:

```
{
     "circumferenceFromDiameter" : function(diameter) {
          return diameter * Math.PI
     },
     "circumferenceFromRadius" : function(radius) {
          return radius * 2 * Math.PI
     }
}
```

After MyFunctions.json is imported into a project, the following expression uses the JSON file's `circumferenceFromDiameter()` function to return the circumference of a circle of diameter 10:

```
footage("MyFunctions.json").sourceData.circumferenceFromDiameter(10)
```

Here is a similar expression using the `circumferenceFromRadius()` function for a circle of radius 5 (both expressions will return the same value):

```
footage("MyFunctions.json").sourceData.circumferenceFromRadius(5)
```

In a more complex expression where you reference multiple functions from the same JSON file, you may want to simplify by loading the path to the JSON data as a variable and refer to it to call the functions, like this:

```
var F = footage("MyFunctions.json").sourceData;
var circleA = F.circumferenceFromDiameter(10);
var circleB = F.circumferenceFromRadius(5);
```

Functions in a JSON file can of course also use After Effects expression code, such as this function which uses the `createPath()` expression method to create a square path centered in the layer:

```
{
     "centerSquare" : function(size) {
          var centerW = thisLayer.width/2;
          var centerH = thisLayer.height/2;
          var radius = size/2;
          var myPoints = [[centerW - radius, centerH + radius], [centerW + radius, centerH + radius],
[centerW + radius, centerH - radius], [centerW - radius, centerH - radius]];

          return createPath(myPoints, [], [] , true)
     }
}
```

You can then apply this expression to the Mask Path property of a layer mask to create a square mask of 250 pixels on each side:

```
footage("MyFunctions.json").sourceData.centerSquare(250)
```

Expression access to path points on masks, Bezier shapes, and brush strokes

You can now use expressions to read and write the x and y coordinates of path points, or *vertices*, for:

- layer masks
- Bezier shapes
- brush strokes on the Paint and Roto Brush & Refine Edge effects.

The new expression methods are similar to accessing path vertices via scripting. The expression method is named `points()` instead of `vertices()` because "points" is a more familiar term for people who don't have a technical background in After Effects scripting.

After Effects also includes a new scriptUI panel, Create Nulls From Paths, which uses these expressions methods so that you don't have to write the expressions yourself. Read more about this scriptUI panel in the "Create Nulls From Paths scriptUI panel" section, below.

The new path point expression methods are:

- `points(t = time)`
- `inTangents(t = time)`
- `outTangents(t = time)`
- `isClosed()`
- `pointOnPath(percentage = 0.5, t = time)`
- `tangentOnPath(percentage = 0.5, t = time)`
- `normalOnPath(percentage = 0.5, t = time)`
- `createPath(points = [[0,0], [100,0], [100,100], [0,100]], inTangents = [], outTangents = [], is_closed = true)`

These are expression methods of the following path objects:

- Layer mask paths. Example path: `layer("Dark Gray Solid 1").mask("Mask 1").path or .maskPath`
- Bezier shape paths. Example path: `layer("Shape Layer 1").content("Shape 1").content("Path 1").path`
- Paint effect stroke paths. Example path: `layer("Dark Gray Solid 1").effect("Paint").stroke("Brush 1").path`
- Roto Brush & Refine Edge effect stroke paths. Example path: `layer("Dark Gray Solid 1").effect("Roto Brush & Refine Edge").stroke("Foreground 1").path`

Tips for working with these expression methods:

- Point and tangent values are returned as an array of [x,y] number pair arrays for the coordinates of the vertices. Coordinate values are rounded to four decimal places.
- Point and tangent arrays start with the first vertex on the path. You can change the first vertex of layer mask paths and Bezier shape paths using the Layer > Mask and Shape Path > Set First Vertex command.
- Coordinates for layer mask path points are relative to the layer's origin in its upper-left hand corner.
- Coordinates for Bezier shape path points are are relative to the anchor point of the path's shape group (ex., "Transform: Shape 1 > Anchor Point").
- Coordinates for brush stroke path points are relative to the start of the stroke; the first point is [0,0].
- The `toComp()` method, or other layer space transform methods, can be useful to convert coordinates from different layers into a common space or simply return the coordinates relative to the composition space.
- Because Bezier shape paths are relative to the anchor point of their local shape group and not the layer, their values may be offset from where you expect them to be. To work around this, set all Transform properties in the shape group to zero. Keep in mind that converting a parametric shape to a Bezier shape may affect the shape group's position.
- The `createPath()` method can be passed points, tangents, and isClosed methods of the same path or other paths. You can pass these methods unchanged to duplicate a path, or transform the point and tangent values before passing them to modify or animate the path.

Expression reference

Path points() method

`{pathProperty}.points(t = time)`

Description

Get the x,y coordinates of all points on a path.

Coordinates for layer mask path points are relative to the layer's origin in its upper-left hand corner.

Coordinates for Bezier shape path points are are relative to the anchor point of the path's shape group (ex., "Transform: Shape 1 > Anchor Point").

Coordinates for brush stroke path points are relative to the start of the stroke; the first point is [0,0].

Optionally specify the time at which sample to the path.

This method can be passed into the `createPath()` method for the `points` parameter when duplicating a path.

Parameters

t	Number, optional. The composition time (in seconds) at which to sample the path. Default is `time` (the current time).

Returns

Array of number pair arrays, rounded to the fourth decimal place.

Path inTangents() method

`{pathProperty}.inTangents(t = time)`

Description

Get the x,y coordinates of the incoming tangent handle for all points on a path.

Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the incoming tangent.

This method can be passed into the `createPath()` method for the `inTangents` parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Parameters

t	Number, optional. The composition time (in seconds) at which to sample the path. Default is `time` (the current time).

Returns

Array of number pair arrays, rounded to the fourth decimal place.

Path outTangents() method

`{pathProperty}.outTangents(t = time)`

Description

Get the x,y coordinates of the outgoing tangent handle for all points on a path.

Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the outgoing tangent.

This method can be passed into the `createPath()` method for the `outTangents` parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Parameters

t	Number, optional. The composition time (in seconds) at which to sample the path. Default is `time` (the current time).

Returns

Array of number pair arrays, rounded to the fourth decimal place.

Path isClosed() method

`{pathProperty}.isClosed()`

Description

Determines if the path is open or closed. Returns true if the path is closed, false if the path is open.

This method can be passed into the `createPath()` method for the `is_closed` parameter when duplicating a path.

Parameters

None.

Returns

Boolean.

Path pointOnPath() method

{pathProperty}`.pointOnPath(percentage = 0.5, t = time)`

Description

Get the x,y coordinates of an arbitrary point along a path.

The point is expressed as a percentage of the arc-length of the path. 0% is the first point and 100% is the last point. When the path is closed, 0% and 100% will return the same coordinates.

Percentage of arc-length is used to ensure uniform speed along the path. Other than 0% and 100%, percentages do not necessarily correlate with the Bezier points on the path. (i.e., For a path with three points, the second point will not necessarily be at 50%.) This also means that for an open path and closed path with identical points, the percentage along the open path will not return the same coordinates as the closed path due to the additional length of the closed path.

Optionally specify the time at which sample to the path.

Parameters

percentage	Number between 0 and 1, optional. The percentage along the arc-length of the path to sample. Values smaller than 0 and larger than 1 are clipped. Default is 0.5.
t	Number, optional. The composition time (in seconds) at which to sample the path. Default is `time` (the current time).

Returns

A number pair array.

Path tangentOnPath() method

{pathProperty}`.tangentOnPath(percentage = 0.5, t = time)`

Description

Get the calculated x,y coordinates of the outgoing tangent handle for an arbitrary point along a path.

Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the outgoing tangent.

The incoming tangent handle is the inverse of this value (multiply the x,y coordinates by -1).

The tangent's parent point is expressed as a percentage of the arc-length of the path. Read the description of the `pointOnPath()` method for details about arc-length percentage.

The coordinates returned by `tangentOnPath()` are *calcuated* from it's parent point and will differ from those returned by `outTangents()` if a user-defined point also exists at that arc-length pecentage. The linear distance between the parent point's coordinates and `tangentOnPath()` c oordinates will always be 1. You can multiply the returned coordinates to create a longer tangent, ex. (`myPath.tangentOnPath() * 100`).

Optionally specify the time at which sample to the path.

Parameters

percentage	Number between 0 and 1, optional. The percentage along the arc-length of the path to sample. Values smaller than 0 and larger than 1 are clipped. Default is 0.5.
t	Number, optional. The composition time (in seconds) at which to sample the path. Default is `time` (the current time).

Returns

A number pair array.

Path normalOnPath() method

`{pathProperty}.normalOnPath(percentage = 0.5, t = time)`

Description

Get the calculated x,y coordinates of the normal for an arbitrary point along a path.

Coordinate values of normals are offset relative to the parent point's coordinates. i.e., The value [0,0] is the same as the parent point.

The normal's parent point is expressed as a percentage of the arc-length of the path. Read the description of the `pointOnPath()` method for details about arc-length percentage.

The coordinates returned by `normalOnPath()` are *calcuated* from its parent point. The linear distance between the parent point's coordinates and `normalOnPath()` coordinates will always be 1. You can multiply the returned coordinates to create a longer normal, ex. (`myPath.normalOnPath() * 100`).

Optionally specify the time at which sample to the path.

Parameters

percentage	Number between 0 and 1, optional. The percentage along the arc-length of the path to sample. Values smaller than 0 and larger than 1 are clipped. Default is 0.5.
t	Number, optional. The composition time (in seconds) at which to sample the path. Default is `time` (the current time).

Returns

A number pair array.

Path createPath() method

`{pathProperty}.createPath(points = [[0,0], [100,0], [100,100], [0,100]], inTangents = [], outTangents = [], is_closed = true)`

Description

Creates a path object from a set of points and tangents.

The points are defined by an array of number pair arrays representing their x,y coordinates. The array length must be at least 1, and can be of any greater length.

The incoming and outgoing tangent handles of the points are defined by an array of number pair arrays representing their x,y offset coordinates. The length of the tangent arrays must be exactly the same as the `points` parameter. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the incoming tangent.

The `points()`, `inTangents()`, `outTangents()`, and `isClosed()` methods of a path can be passed into the *points*, *inTangents*, *outTangents*, and *is_closed* parameters to duplicate a path.

The points and tangents of the same path can be passed into `createPath()` with modifications to generate a different result. For example, the following expression will remove curves from Mask 1 by not passing the inTangents or outTangents parameters:

```
myMask = mask("Mask 1").path;
myMask.createPath(myMask.points());
```

The following example passes the points and tangents of Mask 1 and converts it to an open path by setting `is_closed` to false:

```
myMask = mask("Mask 1").path;
myMask.createPath(myMask.points(), myMask.inTangents(), myMask.outTangents(), false);
```

Parameters

points	An array of length 1 or greater containing number pair arrays representing the [x,y] coordinates of the path points. Required unless no parameters are passed (i.e., `createPath()`). Default is [[0,0], [100,0], [100,100], [0,100]].
inTangents	An array containing number pair arrays representing the [x,y] offset coordinates of the incoming tangent handles to the path points. Required unless no parameters are passed (i.e., `createPath()`). The array length must be the same as *points*, or you can pass an empty array ([]), which will assume the same length as *points* and [0,0] for all tangents. Default is an empty array.
outTangents	An array containing number pair arrays representing the [x,y] offset coordinates of the outgoing tangent handles to the path points. Required unless no parameters are passed (i.e., `createPath()`). The array length must be the same as *points*, or you can pass an empty array ([]), which will assume the same length as *points* and [0,0] for all tangents. Default is an empty array.

is_closed	Boolean, optional. Determines if the mask is closed. If true, the last point will be connected to the first point. Default is true.

Returns

An path object.

Examples

Example 1

Example 1 writes the list of point and tangent coordinates from Path 1 of Shape 1 on layer Shape Layer 1, at time=0, into a string. Apply this to the source text property of a text layer for a readout of the coordinates and incoming and outgoing tangents of the shape.

```
pointsList = "";
sampleTime = 0;
myShape = thisComp.layer("Shape Layer 1").content("Shape 1").content("Path 1").path;

for (i = 0; i < myShape.points(sampleTime).length; i++) {
    pointsList += "c: " + myShape.points(sampleTime)[i].toString() + "  i: " + myShape.inTangents(samp
leTime)[i].toString() + " o: " + myShape.outTangents(sampleTime)[i].toString() + "\n";
}

pointsList;
```

Example 2

Example 2 reads the coordinates of the first vertex of Mask 1 on Dark Gray Solid 1 and converts them to composition coordinates. Apply this to a 2D point control of an effect, such as Write-on or CC Particle Systems II, to make the effect trace or track the first point of an animated mask. Duplicate the effect and change the path points index value ([0]) to trace or track the other points of the mask.

```
myLayer = thisComp.layer("Dark Gray Solid 1");
myLayer.toComp(myLayer.mask("Mask 1").maskPath.points()[0]);
```

Create Nulls From Paths scriptUI panel

Create Nulls From Paths is a scriptUI panel included with After Effects that creates nulls for each point on a mask path or Bezier shape path. These nulls can either control the points or be controlled by the points. The script can also create a null that traces the path. This script automates linking the nulls using the new expression access to path points, so that you don't have to write the expressions yourself.

To open the panel, choose Window > Create Nulls From Paths.jsx.

To create nulls for the points on a path, first select a mask path or Bezier shape path in the Timeline panel, then click one of the buttons in the Create Nulls From Paths panel:

- **Points Follow Nulls** creates *nulls that control* the positions of the path points.
- **Nulls Follow Points** creates *nulls that are controlled by* the positions of the path points.
- **Trace Path** creates a single null with its position linked to coordinates of the path. The null's rotation is linked to auto-orient along the path. A custom effect applied to the null, Trace Path, controls the progress of the null along the path and whether it loops. By default, keyframes are set to trace the path in 1 second, and loop is enabled.

For shape layers, this script and the new expression access to path points only works with Bezier shapes. Parametric shapes (rectangle, ellipse, star, etc.) must be converted to a Bezier shape to work with this script: expand the shape layer contents, right-click on the parametric path (ex., Rectangle 1), and choose Convert To Bezier Path.

Note that the Transform properties of a Bezier shape path or a shape group will cause an offset when the nulls' positions are calculated. In particular, converting a parametric shape to a Bezier shape adjusts the shape's Position property, and will cause such an offset. To avoid this, set the value of the shape's Transform properties (ex., "Transform: Rectangle 1") to zeroes prior to creating the nulls. If necessary, compensate by adjusting the shape layer's Transform properties, instead of the Transform properties of the individual shape paths or groups.

Motion Graphics templates scripting access

Scripts can now add properties to the Essential Graphics panel and export a Motion Graphics template, using the following scripting methods:

Property canAddToMotionGraphicsTemplate() method

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

Description

Test whether or not the property can be added to the Essential Graphics panel for the specified composition. Returns true if the property can be added, false otherwise.

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

Supported property types are:

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

Parameters

comp	The composition that you wish to test adding the property to, compItem. Required.

Returns

Boolean.

Property addToMotionGraphicsTemplate() method

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

Description

Add the property to the Essential Graphics panel for the specified composition. Returns true if the property is successfully added, false otherwise.

If the property is not added, it is either because it is not one of the supported property types or the property has already been added to that composition. After Effects will present a warning dialog. Use the canAddToMotionGraphicsTemplate() method to test whether the property can be added to a Motion Graphics template.

Parameters

comp	The composition that you wish to add the property to, compItem. Required.

Returns

Boolean.

CompItem motionGraphicsTemplateName attribute

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

Description

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

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

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

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

Type

String; read/write.

CompItem exportAsMotionGraphicsTemplate() method

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

Description

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

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

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

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

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

Parameters

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

Returns

Boolean.

CompItem openInEssentialGraphics() method

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

Description

Open the composition in the Essential Graphics panel.

Parameters

None.

Returns

Nothing.

Scripting bug fixes

The following bugs related to scripting are fixed in the October 2017 release of After Effects CC (version 15.0).

- Drop-down menus in scripts no longer fail to display items when the list is very long.
- Drop-down menus in scripts no longer overlap the last item of the list with the menu control if the list is displayed above the control.
- The Scale Composition.jsx script included with After Effects no longer fails if there are locked layers in the composition. Locked layers will now be unlocked, scaled, and re-locked by the script.
- Forcing the command line renderer (aerender) to stop now also forces its aerendercore child processes to stop immediately. Previously, aerendercore processes were allowed to finish their current render task, but that sometimes didn't happen if aerendercore was waiting for a message from the now-stopped aerender.