# Colorway Python API Examples

Last update on September 18th, 2018

# Contents

# Chapter 1

# Colorway Python API Examples

## 1.1 Content

## 1.2 How to use this documentation

This documentation presents a very simple set of global wrapper functions that can be written to accomplish specific tasks. If the functionality of the presented examples is sufficient, feel free to just copy the code verbatim. Otherwise use the example code as a guide to write Your own wrapper functions or classes.

## 1.3 How to create new Colorway project

Before You can start adding images, text, palettes and other Colorway assets You need to have a Colorway project put them in. This section contains an example wrapper function that creates a new Colorway project from scratch. Please refer to: createNewProject for the code.

## 1.4 How to open existing Colorway project

This section contains an example wrapper function that can be used to open an existing Colorway project. Please refer to: openProject for the code.

## 1.5 How to add and setup a new sheet

Before we can add any visible items to Colorway project such as images, text, palettes and other Colorway assets, we need to add at least one sheet and specify it's size. Colorway project can have as many sheets as is required. Following example: addSheet shows how to add a new sheet to a project.

If You need to get the document handle required by this function please refer to section How to find and manipulate existing item in a project, for more information about retrieving handles to existing items in Your project.

Once You have the sheet item's asset You can set it's size using following code, in this example we set the sheet to have the size of a High Definition screen:

```
sheet.size = ( 1920, 1080 );
```

The `document` asset, required by addSheet can be taken from the dictionary returned by createNewProject, by using following code:

```
projectInfo = createNewProject( 'ProjectName', 'MainDocumentName' );
document    = projectInfo[ 'document' ];
```

It can also be taken from a project asset returned by openProject by using following code:

```
projectInfo = openProject( pathToExisingProjet );
document    = projectInfo[ 'document' ];
```

## 1.6 How to add and manipulate various types of items

This section contains examples that show how to create and manipulate various items that can be placed on a sheet.

In Colorway there are following items that can be placed on a sheet:

- contact sheet items, ( see Add a new contact sheet item),
- DCI items, (see Add a new DCI item),
- image items, (see Add a new image item),
- palette items, (see Add a new palette item),
- table items, (see Add a new dynamic table item),
- text items, (see Add a new text item).

Each Colorway item must be placed on some parent sheet, to learn how to set the position, size and rotation of Colorway items refer to section: How to set the position, size and rotation of a Colorway item..

### 1.6.1 Add a new contact sheet item

Contact sheet items allow for displaying of different variations of items, groups or entire sheets, automatically laid-out as a grid. They are great for presenting, in a compact form, different color schemes, palettes or material assignments of DCI items.

Following example: addContactSheetItem shows how to add a new contact sheet item.

### 1.6.2 Add a new DCI item

DCI (Deep Color Image) items, refer to special Colorway assets that represent 2D renders of 3D objects. DCI items contain various render passes that allow to modify certain aspects of the object's appearance without the need for time consuming re-renders. Most notably, the DCI items allow to change the colors of predefined parts of 3D object, and to change the colors and intensities of lights in the 3D scene. Additionally, You can assign materials or individual textures to the predefined parts.

Please refer to following sections for more information on manipulating the appearance of DCI items:

- To see how to change colors of parts of a DCI item see: How to define new palettes and apply colors to items
- To see how to apply materials to parts of a DCI item see: How to assign and manipulate SubMaterials
- To see how to apply individual textures to parts of a DCI item see: How to add texture layers

Following example: addDciItem shows how to add a new DCI item.

### 1.6.3 Add a new image item

Image items represent 2D raster or vector images. Each image item, can have a list of image files (assets) it can refer to as variants. For each variation, image item can have a different variant selected as active one.

Vector images in the SVG format allow specific regions of the image to be defined as parts, similar to DCI items (see Add a new DCI item).

Following example: addImageItem shows how to add a new image item.

### 1.6.4 Add a new palette item

Palette items are visual representations of palettes.

Following example: addPaletteItem shows how to add a new palette item, to see a graphical representation of a palette on canvas.

### 1.6.5 Add a new dynamic table item

Dynamic table items are used to display various information about other Colorway items in a form of a table. Usually dynamic tables are used to display the parts, colors, materials, and meta-data of a DCI item.

Following example: addDynamicTableItem shows how to create a new dynamic table item.

### 1.6.6 Add a new text item

Text item's represent blocks of text. There are two main types of text items used in Colorway: local and global.

Local text items store the text in their state, which is different for each variation of the parent sheet.

Global text items store the text as a variant. Global text items, can have multiple variants, and each variation of the parent sheet can select the specific variant to be the active one in it.

Following example: addLocalTextItem shows how to create a local text item.

Following example: addGlobalTextItem shows how to create a global text item.

### 1.6.7 How to set the position, size and rotation of a Colorway item.

When Colorway item is created and added to the parent sheet or group, by default all aspects of it's placement on the sheet are reset to zero: it's position, size and rotation. This section contains examples showing how to set specific aspect of item's placement using individual helper functions, or all at once using one helper function.

Following example: setItemPosition shows how to set item's position.

Following example: setItemSize shows how to set item's size.

Following example: setItemRotation shows how to set item's rotation.

Following example: setWorldTransform shows how to set item's world transform matrix.

### 1.6.8 Change the font family, font style and font size of Colorway item's with text.

A number of colorway items such as: text items, palette items, contact sheet items and table items, display text as part of their visual representation. They allow to configure the font that is used to display that text.

Following example: setFontItemStyle shows how to set-up item's font, it's size and other parameters.

## 1.7 How to find and manipulate existing item in a project

When working on a complex project sometimes we need to retrieve resource handles of different items to manipulate them, for example, to change their position. To do that we can browse the the main directory of the Colorway project entity.

This documentation presents a simple function to get a list of resource getResourcesOfType shows how to retrieve a list of resource handles of different types from a given directory.

If we needed to retrieve all DCI items from a project we could use following code:

```
projectInfo              = openProject( pathToExisingProjet );
projectResourceDirecotry = projectInfo[ 'projectDirectoryHandle' ];
listOfDCIs               = getResourcesOfType( projectResourceDirecotry, 'dci' );
```

## 1.8 How to add new variants and select the active one

Most of Colorway items, that represent various assets like images, text or 3D objects, allow to store multiple assets within one entity. The assets stored within a single Colorway item are called variants. In a given variation, each item can have only one variant selected as active.

Following example: addLinkAsItemVariant shows how to add a new variant to an item.

Following example: createLink shows how to create a link to some asset, that can later be added to an item.

Following example: getItemVariants shows how to get a list of available variants in a given Colorway item.

Following example: setResourceHandleAsActiveVariant shows how to select specific variant of an item as active one for specific variation.

## 1.9 How to add/remove variations

Variations allow to take a snapshot of the current state of a sheet and store it. Each stored variation can be recalled by selecting it as active, or manipulated directly.

You can get current working variation of any sheet by executing following code:

```
currentVariation = sheet.workingVariation;
```

If You want a specific variation to become the current (selected) one, execute following code:

```
sheet.selectedVariation = variationToBeSelected;
```

### 1.9.1 Add new variation to the given sheet

Following example: addVariation shows how to add new variation to the given sheet.

### 1.9.2 Remove specific variation from the given sheet

Following example: removeVariation shows how to remove specific variation from the given sheet.

## 1.10 How to define new palettes and apply colors to items

Palettes are collections of swatches which in turn represent individual colors. Each sheet can have a different palette selected as active for each of it's variations. Within a palette each swatch has a numerical index assigned to it. When assigning colors to different. When assigning colors to Colorway items, like for example DCI item, we can create a mapping between the item (or item's part) and an index within a palette. By switching palette, we can then generate completely different color scheme for that item. We can also create swatches not assigned to any palette. When such an independent swatch is assigned to anything, it's color will remain constant, even if the active palette is changed.

Following example: addPalette shows how to create a new palette to store colors in.

Following example: addSwatchToPalette shows how to add swatches (colors) to existing palette.

Following example: createStaticSwatch shows how to create color swatch that is not a part of any palette, and therefore will maintain it's color even if active palette is changed.

Following example: createDynamicSwatch shows how to create dynamic swatch, referencing specific index in the active palette.

Following example: setObjectSwatch shows how to assign a color to an object.

Following example: applyPartNameToSwatchMapping shows how to assign swatches to multiple parts by creating a dictionary.

## 1.11 How to assign and manipulate SubMaterials

Colorway allows for specifying material libraries. Material libraries store predefined materials, that can be applied to parts of multi-part Colorway items such as vector image items and DCI items.

To get a list of available materials You first need to create a resource handle representing the directory of the material library, using following code:

```
from colorway.livesource import ResourceHandle;
materialLibraryResourceHande = ResourceHandle( '/matLib' );
```

Once we have the resource handle of the material library directory, we can get all available materials by using the example function getResourcesOfType (refer to section How to find and manipulate existing item in a project for more details ).

For example:

```
materialResourceHandles = getResourcesOfType( materialLibraryResourceHande, 'material' );
```

Following example: applySubMaterialToPart shows how to apply sub-material to a part of multi-part item. The function requires the we know the id number of the part we want the material to be applied to. We can get a dictionary, mapping part names to their id number by using the example function getPartNameToIdMap.

Let us assume we already have a DCI item stored in variable `dciItem` and it's parent sheet stored in variable `sheet`. Let us also assume that the DCI item has some part named `'Part1'` and that we want to apply the first material from the material library to that part. Provided that we have implemented the example wrapper API presenting in this documentation, we can achieve that by executing following code:

```
dciItemState = getItemWorkingState( dciItem, sheet.workingVariation );
partNameToIdMap = getPartNameToIdMap( dciItemState );
applySubMaterialToPart( dciItemState, partNameToIdMap[ 'Part1' ], materialResourceHandles[0] );
```

## 1.12 How to add texture layers

Colorway allows to apply images to parts of multi-part items. Images applied in such a way are called texture layers.

Following example addTextureLayerToPart shows how to add a new texture layer to part of a multi-part item.

## 1.13 How to save a Colorway project

By default all newly created Colorway project are stored in a temporary location that is going to be erased as soon as the projects are closed. To prevent that, You need to specify a more permanent location to the projects in. This section contains examples of wrapper function that can be used to save given Colorway project to specific location, and then to simply save changes made to a project, which already has a specific storage location.

### 1.13.1 Save changes to Colorway project already saved to specific location

Following example: saveProjectAs shows how to save project to specific location.

### 1.13.2 Save Colorway project to specific location

Following example: saveProject shows how to save changes to a project that already has a specific storage location.

## 1.14 How to export Colorway project

Exporting a Colorway project allows to capture what is seen on the canvas and store it in a format more suitable for exchanging between different applications than Colorway, for example a PNG image or a PDF document.

Colorway offers three different ways that variations can be exported:

- as complete, composed images,
- as individual layers,
- as source images.

Complete, composed images look exactly as what You can see on the canvas. Individual layers, store each Colorway item as a separate image scaled to the size of parent sheet for easy compositing in different application. Source images contain only the individual items, cropped.

### 1.14.1 Export variations as PDF document[s]

Following example: exportVariationsAsPdf shows how to export a given list of variations as a single or multiple PDF documents.

### 1.14.2 Export variations as individual images in PNG format

Following example: exportVariationsAsIndividualPngs shows how to export a given list of variations as a set of individual images in PNG format.

### 1.14.3 Export variations as layers in PNG format

Following example: exportVariationsAsPngLayers shows how to export a given list of variations as a set of images where each item is an individual PNG image scaled to the size of the parent sheet. The images are prepared in a way that can be easily loaded to a graphical program as layers for compositing.

### 1.14.4 Export variations as source images in PNG format

Following example: exportVariationsAsPngSources shows how to export a given list of variations as a set of images where each item is an individual PNG image in its original size.

## 1.15 How to consolidate Colorway project

Consolidation allows a Colorway project that uses assets that are taken from sources that lay outside the main directory of the Colorway project itself, to be packed in an self-containing archive that can be moved to a new location, where those sources might not be available.

Following example: consolidateProject shows how to consolidate Colorway project.

# Chapter 2

# Namespace Index

## 2.1 Packages

Here are the packages with brief descriptions (if available):

# Chapter 3

# Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 4

# Namespace Documentation

## 4.1 ColorwayPythonAPIExamples Namespace Reference

The namespace containing all the example functions, classes and variables.

### Classes

- class ColorAssignmentPoint

  *Allowed values for colorAssignmentPoint parameter.*

### Functions

- def addContactSheetItem (name, parent, referencedItem)

  *Add a new contact sheet item go given sheet or group.*
- def addDciItem (name, stringId, parent, variation)

  *Add a new DCI item to a given sheet or group.*
- def addDynamicTableItem (name, parent, baseItemReference)

  *Add a new table item to given sheet or group.*
- def addGlobalTextItem (name, text, parent)

  *Add a new global text item to give sheet or group.*
- def addImageItem (name, stringId, parent, variation)

  *Add a new image item to given sheet or group.*
- def addLinkAsItemVariant (item, name, linkTargetStringIdentifier)

  *Add a new link variant to given item.*
- def addLocalTextItem (name, text, parent)

  *Add a new local text item to given sheet or group.*
- def addPalette (directoryResource, paletteName)

  *Add new palette.*
- def addPaletteItem (name, paletteStringIdentifier, parent, variation)

  *Add new palette item to a given sheet or group.*
- def addSheet (name, document)

  *Add a new sheet to the document.*
- def addSwatchToPalette (paletteResource, swatchName, color)

  *Add new swatch to a given palette.*

- def addTextureLayerToPart (multiPartItemState, partId, textureStringIdentifier)

  *Add texture layer to a part.*
- def addVariation (name, sheet)

  *Add a new variation to given sheet.*
- def applyPartNameToSwatchMapping (multiPartItemState, partNameToSwatchMap, partNameToIdMap)

  *Apply part name to swatch index mapping for a multi-part item.*
- def applySubMaterialToPart (multiPartItemState, partId, materialHandle)

  *Apply submaterial to part of a multi-part item.*
- def checkVersionAndBuildDate ()

  *Check Colorway version and build date.*
- def consolidateProject (project, targetPath, typesToConsolidate)

  *Consolidate a Colorway project.*
- def createBaseItemReference (item, propertyName)

  *Create base item reference.*
- def createDynamicSwatch (swatchName, indexInPalette)

  *Create dynamic swatch from active palette.*
- def createLink (name, parentDirectory, linkTargetStringIdentifier)

  *Create a link in `parentDirectory` to a resource identified by `targetStringIdentifier`.*
- def createNewProject (projectName, documentName)

  *Create new Colorway project.*
- def createStaticSwatch (swatchName, color)

  *Create new static swatch for a color.*
- def exportVariationsAsIndividualPngs (variations, document, outputDir, namePrefix)

  *Export list of variations as individual PNG files.*
- def exportVariationsAsPdf (variations, document, outputDir, namePrefix, asSingleFile)

  *Export list of variations in PDF format.*
- def exportVariationsAsPngLayers (variations, document, outputDir, namePrefix)

  *Export list of variations as individual layers in PNG format.*
- def exportVariationsAsPngSources (variations, document, outputDir, namePrefix)

  *Export sources for given variations as individual PNG files.*
- def getItemVariants (item)

  *Get a list of available variants.*
- def getItemWorkingState (item, variation)

  *Get item's state for given item.*
- def getPartNameToIdMap (multiPartItemState)

  *Create a dictionary mapping part's name to its numerical id.*
- def getResourcesOfType (directoryHandle, typeName)

  *Retrieve a list of all assets of specific type available in given project.*
- def openProject (path)

  *Open existing Colorway project.*
- def removeVariation (variation, sheet)

  *Remove given variation form given sheet.*
- def saveProject (project)

  *Save changes to a Colorway project.*
- def saveProjectAs (project, path)

  *Save Colorway project to location.*
- def selectActivePalette (sheet, paletteH)

  *Select active palette for a sheet.*
- def setFontItemStyle (fontItem, fontFamily, fontPointSize, isBold, isItalic, isUnderline)

  *Set the style of an item that uses a font.*
- def setItemPosition (item, x, y)

*Set the global position of a given item: Following example shows a simple function that can be used to set the global position of an item.*

- def setItemRotation (item, angle)

  *Set the rotation of a given item.*

- def setItemSize (item, width, height)

  *Set the size of a given item: The following example shows a simple function that can be used to set the size of an item.*

- def setObjectSwatch (objectWithColor, sheet, assignmentPoint, swatch)

  *Assign color to an object.*

- def setResourceHandleAsActiveVariant (item, resourceHandle, variation)

  *Set resource handle as active variant of an item.*

- def setWorldTransform (itemState, positionX, positionY, pivotX, pivotY, rotation, scaleX, scaleY)

  *Set the world position, rotation and scale for a given item.*

## Variables

- list TypesToConsolidate

  *List of types to include when consolidating a project.*

### 4.1.1 Detailed Description

The namespace containing all the example functions, classes and variables.

This namespace contains all the example functions, variables and classes described in this documentation. If You find the functionality implemented here sufficient to achieve Your goals, feel free to just copy the code verbatim. Otherwise You can use the examples as guides to implement You own wrapper function of classes, with functionality more suited to Your needs.

### 4.1.2 Function Documentation

#### 4.1.2.1 addContactSheetItem()

```
def ColorwayPythonAPIExamples.addContactSheetItem (
              name,
              parent,
              referencedItem )
```

Add a new contact sheet item go given sheet or group.

Contact sheet items allow for presenting, in a form of a grid, different variations of an item, group of items or entire sheets.

**Parameters**

| | |
|---|---|
| *name* | Name of the newly created contact sheet item. |
| *parent* | Parent sheet or group the new contact sheet item is to be added to. |
| *referencedItem* | The item, variations of which the contact sheet should display. Usually a sheet item. |

**Returns**

Newly created contact sheet item.

**Warning**

The newly created contact sheet item has no size, and will not be displayed. See setItemSize for information on how to specify the size of an item.

**See also**

setItemSize
setItemPosition
setItemRotation
How to add and manipulate various types of items

```
617 def addContactSheetItem( name, parent, referencedItem ):
618     from colorway.canvas import ContactSheetItem;
619     item               = ContactSheetItem( parent.owningDocument );
620     item.name          = name;
621     item.identifier    = colorway.canvas.operations.call( 'findMaxItemId', parent.owningDocument ) + 1;
622     item.referencedItem = referencedItem;
623     item.call( 'addToParent', parent );
624     return item;
625
```

#### 4.1.2.2 addDciItem()

```
def ColorwayPythonAPIExamples.addDciItem (
              name,
              stringId,
              parent,
              variation )
```

Add a new DCI item to a given sheet or group.

DCI (Deep Color Image) items are special Colorway assets that represent 2D renders of 3D objects. DCI Items allow to modify the appearance of the object's appearance without the need for re-renders.

Adding a new item to a sheet or group makes it appear in all variations of that specific sheet. However, each variation can have a different variant selected as active for each item. For more information on how to add and select variants, please refer to section: How to add new variants and select the active one.

Creating a new DCI item consists of three steps:

- creating the DCI item and assigning it to parent (sheet or group),
- creating a link to the DCI file (asset) we want the item to refer to, and adding it to the list of it's variants,
- selecting the newly created link as the active one in the new item's state for a given variation.

Following example shows a simple function that can be used to add a new DCI item to a sheet or a group. To make the code simpler and more modular we extracted the step of creating a link to specific file to a separate utility function: addLinkAsItemVariant.

**Parameters**

| | |
|---|---|
| *name* | Name of the new newly created DCI item. |
| *stringId* | Path to the DCI file we want to load. |
| *parent* | Parent sheet or group, the new DCI item is to be added to. |
| *variation* | Variation the specific DCI file (asset) is to be active in. |

**Returns**

Newly created DCI item.

**Warning**

The newly DCI item has no size, and will not be displayed. See setItemSize for information on how to specify the size of an item.

**See also**

addLinkAsItemVariant
setItemSize
setItemPosition
setItemRotation
How to add new variants and select the active one
How to add and manipulate various types of items

```
660 def addDciItem( name, stringId, parent, variation ):
661     from colorway.canvas     import DciItem;
662     item            = DciItem( parent.owningDocument );
663     item.name       = name;
664     item.identifier = colorway.canvas.operations.call( 'findMaxItemId', parent.owningDocument ) + 1;
665     item.call( 'addToParent', parent );
666 
667     resource = addLinkAsItemVariant( item, name, stringId );
668 
669     itemState                = variation.call( 'getStateFor', item );
670     itemState.source.variant = ResourceHandle( resource );
671     return item;
672 
```

#### 4.1.2.3 addDynamicTableItem()

```
def ColorwayPythonAPIExamples.addDynamicTableItem (
              name,
              parent,
              baseItemReference )
```

Add a new table item to given sheet or group.

Following example shows a simple function that creates a new dynamic table item.

**Parameters**

| | |
|---|---|
| *name* | Name of the newly created dynamic table item. |
| *parent* | Parent sheet or group, the new item is to be added to. |
| *baseItemReference* | Reference object representing, an aspect of an item the table item is to display (see createBaseItemReference ). |

**Returns**

Newly created dynamic table item.

**See also**

createBaseItemReference
Add a new dynamic table item

```
825 def addDynamicTableItem( name, parent, baseItemReference ):
826     from colorway.canvas import TableItem;
827     item                  = TableItem( parent.owningDocument );
828     item.name             = name;
829     item.identifier       = colorway.canvas.operations.call( 'findMaxItemId', parent.owningDocument ) + 1;
830     item.propertyReference = baseItemReference;
831     item.call( 'addToParent', parent );
832     return item;
833
```

#### 4.1.2.4 addGlobalTextItem()

```
def ColorwayPythonAPIExamples.addGlobalTextItem (
              name,
              text,
              parent )
```

Add a new global text item to give sheet or group.

Following example shows a simple function that creates a new global text item.

**Parameters**

| | |
|---|---|
| *name* | Name of the newly created text item. |
| *text* | Text to be displayed by the text item. |
| *parent* | Parent sheet or group, the new text item is to be added to. |

**Returns**

Newly created text item.

**See also**

Add a new text item

```
868 def addGlobalTextItem( name, text, parent ):
869     from colorway.canvas     import TextItem;
870     item            = TextItem( parent.owningDocument );
871     item.name       = name;
872     item.identifier = colorway.canvas.operations.call( 'findMaxItemId', parent.owningDocument ) + 1;
873     item.call( 'addToParent', parent );
874 
875     from colorway import Text;
876     from colorway.livesource import Resource;
877     asset       = Text();
878     asset.text  = text;
879     itemAssets  = item.embeddedSourceEnsureExists.resource;
880     resource    = Resource.createResource( itemAssets, Resource.getResourceTypeById( 'text' ), asset, name
      + '.txt', True, True );
881 
882     state                = item.owningSheet.workingVariation.call( 'getStateFor', item );
883     state.source.variant = ResourceHandle( resource );
884 
885     return item;
886 
```

#### 4.1.2.5 addImageItem()

```
def ColorwayPythonAPIExamples.addImageItem (
              name,
              stringId,
              parent,
              variation )
```

Add a new image item to given sheet or group.

Image items represent 2D raster or vector images.

Creating a new image item consists of three steps:

- creating the image item and assigning it to the parent (sheet or group),
- creating a link to the image file (asset) we want the item to refer to, and adding it to the list of it's variants,
- selecting the newly created link as the active one in the new item's state for a given variation.

Following example shows a simple function that can be used to add a new image item to a sheet or a group. To make the code simpler and more modular we extracted the step of creating a link to specific file to a separate utility function: addLinkAsItemVariant.

**Parameters**

| | |
|---|---|
| *name* | Name of the newly created text item. |
| *stringId* | Path to the image file we want to load. |
| *parent* | Parent sheet or group, the new text item is to be added to. |
| *variation* | Variation the specific image file (asset) is to be active in. |

**Returns**

The newly created image item.

**Warning**

The newly created image item has no size, and will not be displayed. See setItemSize for information on how to specify the size of an item.

**See also**

addLinkAsItemVariant
setItemSize
setItemPosition
setItemRotation
How to add new variants and select the active one
How to add and manipulate various types of items

```
702 def addImageItem( name, stringId, parent, variation ):
703     from colorway.canvas     import ImageItem;
704     item            = ImageItem( parent.owningDocument );
705     item.name       = name;
706     item.identifier = colorway.canvas.operations.call( 'findMaxItemId', parent.owningDocument ) + 1;
707     item.call( 'addToParent', parent );
708 
709     resource   = addLinkAsItemVariant( item, name, stringId );
710 
711     itemState               = variation.call( 'getStateFor', item );
712     itemState.source.variant = ResourceHandle( resource );
713     return item;
714 
```

#### 4.1.2.6 addLinkAsItemVariant()

```
def ColorwayPythonAPIExamples.addLinkAsItemVariant (
              item,
              name,
              linkTargetStringIdentifier )
```

Add a new link variant to given item.

Following example shows a simple function that creates a link to an asset, and then adds it as item's variant. For example if we have a link to an image file, we can add it as variant to an image item. Colorway items such as: image items, dci items, text items and others, can have multiple variants and each different variant can the active one in different variation.

**Parameters**

| | |
|---|---|
| *item* | The item we want the link to add to. |
| *name* | The name of the link. |
| *linkTargetStringIdentifier* | Path to the asset, the link is to point to. |

**Returns**

Newly created link.

**See also**

createLink
How to add new variants and select the active one

```
945 def addLinkAsItemVariant( item, name, linkTargetStringIdentifier ):
946     item.liveSource = item.embeddedSourceEnsureExists;
947     parentDirectory = item.embeddedSource.resource;
948 
949     return createLink( name, parentDirectory, linkTargetStringIdentifier );
950 
```

#### 4.1.2.7 addLocalTextItem()

```
def ColorwayPythonAPIExamples.addLocalTextItem (
              name,
              text,
              parent )
```

Add a new local text item to given sheet or group.

Following example shows a simple function that creates a new local text time.

**Parameters**

| | |
|---|---|
| *name* | Name of the newly created text item. |
| *text* | Text to be displayed by the text item. |
| *parent* | Parent sheet or group, the new text item is to be added to. |

**Returns**

Newly created text item.

**See also**

addGlobalTextItem
Add a new text item

```
846 def addLocalTextItem( name, text, parent ):
847     from colorway.canvas     import TextItem;
848     item            = TextItem( parent.owningDocument );
849     item.name       = name;
850     item.identifier = colorway.canvas.operations.call( 'findMaxItemId', parent.owningDocument ) + 1;
851     item.call( 'addToParent', parent );
852 
853     state               = item.owningSheet.workingVariation.call( 'getStateFor', item );
854     state.variationText = text;
855 
856     return item;
857 
```

#### 4.1.2.8 addPalette()

```
def ColorwayPythonAPIExamples.addPalette (
              directoryResource,
              paletteName )
```

Add new palette.

Palettes are collections of swatches. The swatches within a palette have numbers assigned to them. When You assign swatches from palettes to items, for example to different parts of a DCI item, this creates a color assignment mapping between parts and indices of the palette. This mapping is not binded to any specific palette, which allows swiching palettes and creating different color schemes based on the same mapping.

The following example shows a simple function that can be used to add a new palette to Colorway project. If you want to define colors for the new palette please refer to addSwatchToPalette.

Palettes must be added to a directory. The best place to store palettes is within the palette directory of the Colorway project.

```
project.palettesDirectory.resource
```

Where `project` variable is a Colorway project created with createNewProject or openProject.

**Parameters**

| | |
|---|---|
| *directoryResource* | Resource refering to the directory, we want the palette to be added to. |
| *paletteName* | Name of the newly added palette. |

**Returns**

Newly created palette.

**See also**

addSwatchToPalette
addPaletteItem
createNewProject
openProject

```
1201 def addPalette( directoryResource, paletteName ):
1202     from colorway import Palette;
1203     from colorway.livesource import Resource;
1204     palette  = Palette();
1205     resource = Resource.createResource( directoryResource, Resource.getResourceTypeById( 'palette' ),
      palette, paletteName + '.jpl', True, True );
1206     return resource;
1207
```

#### 4.1.2.9 addPaletteItem()

```
def ColorwayPythonAPIExamples.addPaletteItem (
                name,
                paletteStringIdentifier,
                parent,
                variation )
```

Add new palette item to a given sheet or group.

Palette items are visual representations of palettes. They can refer to the active palette (dynamic), specific palette from the palettes directory, or external asset file on disk, either in the JPL or the CPL format. Following example shows a simple function that can be used to create a new palette item.

**Parameters**

| | |
|---|---|
| *name* | Name of the new palette item. |
| *variation* | Variation the specific palette (asset) is to be active in. |
| *parent* | Parent sheet or group, the new item is to be added to. |
| *paletteStringIdentifier* | String identifier of the palette resource, the palette item is to represent. |

**Returns**

Newly created palette item.

If `paletteStringIdentifier` parameter is set to `None`, the palette item will represent the active palette of the given variation of the parent sheet.

If You want the palette item to represent a palette created with the example function addPalette, set the `paletteStringIdentifier` paremeter to:

```
'@Palettes/Palette_Name.jpl'
```

Where `Palette_Name` should be substituted with the name assigned to the palete when it was created, and .jpl is the default extension, also assigned to the palette when is was created. See the code of addPalette for more details.

**Warning**

The newly created palette item has no size, and will not be displayed. See setItemSize for information on how to specify the size of an item.

**See also**

addPalette
addSwatchToPalette
createDynamicSwatch
addPaletteItem
How to add and manipulate various types of items
How to define new palettes and apply colors to items

```
749 def addPaletteItem( name, paletteStringIdentifier, parent, variation ):
750     from colorway.canvas import PaletteItem;
751     item            = PaletteItem( parent.owningDocument );
752     item.name       = name;
753     item.identifier = colorway.canvas.operations.call( 'findMaxItemId', parent.owningDocument ) + 1;
754     item.call( 'addToParent', parent );
755 
756     if paletteStringIdentifier:
757         resource    = addLinkAsItemVariant( item, name, paletteStringIdentifier );
758 
759         itemState               = variation.call( 'getStateFor', item );
760         itemState.source.variant = ResourceHandle( resource );
761 
762     return item;
763 
```

#### 4.1.2.10 addSheet()

```
def ColorwayPythonAPIExamples.addSheet (
              name,
              document )
```

Add a new sheet to the document.

Following example shows a simple function that creates a new sheet and adds it to a given document.

**Parameters**

| | |
|---|---|
| *name* | The name of the newly created sheet. |
| *document* | Parent documnet, the new sheet is to be added to. |

The document handle can be obtained when a new Colorway project is being created (please refer to: createNew↩ Project ) or retrieved from an already opened project using the getResourcesOfType helper function. Please refer to section: How to find and manipulate existing item in a project, for more information.

**Returns**

This function returns a newly created sheet item's asset.

**Warning**

The newly sheet has no size, and will not be displayed.

**See also**

createNewProject
How to add and setup a new sheet
How to create new Colorway project
How to open existing Colorway project

```
784 def addSheet( name, document ):
785     from colorway.canvas import SheetItem;
786     sheet            = SheetItem( document );
787     sheet.name       = name;
788     sheet.identifier = colorway.canvas.operations.call( 'findMaxItemId', document ) + 1;
789     sheet.call( 'addToParent', document );
790     return sheet;
791 
```

#### 4.1.2.11 addSwatchToPalette()

```
def ColorwayPythonAPIExamples.addSwatchToPalette (
              paletteResource,
              swatchName,
              color )
```

Add new swatch to a given palette.

The following example shows a simple function that can be used to add new swatch to a palette created by: addPalette.

**Parameters**

| | |
|---|---|
| *paletteResource* | Resource referring to the palette we want the swatch to be added to. |
| *swatchName* | Name of the new swatch. |
| *color* | String identifying the new color in the hexadecimal code. For example: `'#ff0000'` for pure red color. |

**Returns**

Newly created swatch.

**See also**

addPalette
addPaletteItem
selectActivePalette
How to define new palettes and apply colors to items

```
1224 def addSwatchToPalette( paletteResource, swatchName, color ):
1225     from colorway import Swatch;
1226     from colorway.livesource import Resource;
1227     swatch = Swatch();
1228     swatch.color = color;
1229     resource = Resource.createResource( paletteResource, Resource.getResourceTypeById( 'swatch' ), swatch,
      swatchName, True, True );
1230     return resource;
1231
```

#### 4.1.2.12 addTextureLayerToPart()

```
def ColorwayPythonAPIExamples.addTextureLayerToPart (
              multiPartItemState,
              partId,
              textureStringIdentifier )
```

Add texture layer to a part.

Following example shows a simple function that shows how to add a new texture layer to a part of a multi-part item.

**Parameters**

| | |
|---|---|
| *multiPartItemState* | The state of a multi-part item we want the texture layer to be added to. |
| *partId* | Id number of the part the texture layer is to be added to. |
| *textureStringIdentifier* | Path to the image file we want to add as texture layer.. |

**See also**

addLinkAsItemVariant
setResourceHandleAsActiveVariant
How to add texture layers

```
1097 def addTextureLayerToPart( multiPartItemState, partId, textureStringIdentifier ):
1098     materialState  = multiPartItemState.call( 'materialStateForPartId', partId, True );
1099     material       = materialState.liveMaterial;
1100
1101     textureLayer   = material.call( 'createLayer' );
1102     linkResource   = addLinkAsItemVariant( textureLayer, os.path.basename( textureStringIdentifier ),
     textureStringIdentifier );
1103     resourceHandle = ResourceHandle( linkResource );
1104
1105     setResourceHandleAsActiveVariant( textureLayer, resourceHandle, multiPartItemState.owningVariation );
1106
```

#### 4.1.2.13 addVariation()

```
def ColorwayPythonAPIExamples.addVariation (
              name,
              sheet )
```

Add a new variation to given sheet.

Following example shows a simple function that can be used to add a new variation to a given sheet. Creating a new variation freezes the current state of a given sheet, and lets it be recalled later on or displayed in a contact sheet with other variations.

Adding a new variation consists of two steps:

- creating a new variation,
- adding it to the list of variations of the sheet.

**Parameters**

| | |
|---|---|
| *name* | Name of the new variation to be added. |
| *sheet* | Parent sheet, the variation is to be added to. |

**Returns**

Newly created variation.

**See also**

removeVariation
addSheet
Add new variation to the given sheet

```
909 def addVariation( name, sheet ):
910     variation = sheet.call( 'createVariation', name );
911     sheet.call( 'addVariation', variation );
912     return variation;
913
```

#### 4.1.2.14 applyPartNameToSwatchMapping()

```
def ColorwayPythonAPIExamples.applyPartNameToSwatchMapping (
              multiPartItemState,
              partNameToSwatchMap,
              partNameToIdMap )
```

Apply part name to swatch index mapping for a multi-part item.

Following example shows a simple function that can be used to apply colors to multiple parts of a multi-part item.

**Parameters**

| | |
|---|---|
| *multiPartItemState* | The state of a multi-part item we want the swatches to be applied to. |
| *partNameToSwatchMap* | A dictionary mapping the name of a part to the swatch id within a palette. |
| *partNameToIdMap* | A dictionary mapping the name of a part to it's id number (see getPartNameToIdMap). |

The `partNameToSwatchMap` can be constructed in following way:

```
partNameToSwatchMap = { 'PART1', 0,
                        'PART2', 1 };
```

Where `'PART1'` and `'PART2'` are the names of parts in a multi-part item and the `0` and `1` are swatch indices within the active palette.

**See also**

getPartNameToIdMap
How to define new palettes and apply colors to items

```
1059 def applyPartNameToSwatchMapping( multiPartItemState, partNameToSwatchMap, partNameToIdMap ):
1060     for name in partNameToSwatchMap:
1061         materialState = multiPartItemState.call( 'materialStateForPartId', partNameToIdMap[ name ], True );
1062         tintSwatch    = createDynamicSwatch( '', partNameToSwatchMap[ name ] );
1063         setObjectSwatch( materialState.owningObject, multiPartItemState.owningObject.owningSheet,
      ColorAssignmentPoint.TintColor, tintSwatch );
1064
```

#### 4.1.2.15 applySubMaterialToPart()

```
def ColorwayPythonAPIExamples.applySubMaterialToPart (
              multiPartItemState,
              partId,
              materialHandle )
```

Apply submaterial to part of a multi-part item.

Following example shows a simple function that can be used to apply a given material to a specific part of a multi-part item.

**Parameters**

| | |
|---|---|
| *multiPartItemState* | The state of a multi-part item we want the swatches to be applied to. |
| *partId* | Id number of the part the material is to be applied to. |
| *materialHandle* | Handle to the material to be applied. |

**Returns**

Newly created submaterial of the given part.

**See also**

getPartNameToIdMap
How to assign and manipulate SubMaterials

```
1078 def applySubMaterialToPart( multiPartItemState, partId, materialHandle ):
1079     materialState = multiPartItemState.call( 'materialStateForPartId', partId, True );
1080     materialState.liveMaterial.call( 'createSubMaterial' );
1081
1082     lastSubMaterialState = materialState.call( 'subMaterialState', materialState.subMaterialStatesCount - 1
       );
1083     lastSubMaterialState.resourceHandle = materialHandle;
1084     return lastSubMaterialState;
1085
```

#### 4.1.2.16 checkVersionAndBuildDate()

```
def ColorwayPythonAPIExamples.checkVersionAndBuildDate ( )
```

Check Colorway version and build date.

The following example shows a simple function that can be used to get information about the version and build date of Colorway.

```
1169 def checkVersionAndBuildDate():
1170     import colorway
1171     versionTuple          = colorway.version();
1172     versionInfo           = {};
1173     versionInfo['tuple']  = versionTuple;
1174     versionInfo['string'] = "{0!s}.{1!s}v{2!s}".format( *versionTuple );
1175     versionInfo['date']   = colorway.builddate();
1176     return versionInfo;
1177
```

#### 4.1.2.17 consolidateProject()

```
def ColorwayPythonAPIExamples.consolidateProject (
              project,
              targetPath,
              typesToConsolidate )
```

Consolidate a Colorway project.

Following example shows a simple function that can be used to consolidate a project into an independent archive.

**Parameters**

| | |
|---|---|
| *project* | Project to consolidate. |
| *targetPath* | The full path of the archive to be created during consolidation. |
| *typesToConsolidate* | List of types to include when consolidating a project. See TypesToConsolidate. |

**Returns**

String containing the error message, if any errors occurred. Empty string otherwise.

**See also**

TypesToConsolidate
How to consolidate Colorway project

```
587 def consolidateProject( project, targetPath, typesToConsolidate ):
588     from colorway.canvas import Consolidator
589     consolidator = Consolidator();
590     errorMessage = '';
591     consolidator.call( 'consolidate', project, targetPath, True,
592                        typesToConsolidate, 'ConsolidateRepositories', errorMessage );
593     return errorMessage;
594
```

#### 4.1.2.18 createBaseItemReference()

```
def ColorwayPythonAPIExamples.createBaseItemReference (
              item,
              propertyName )
```

Create base item reference.

Following example shows a simple function that creates a base item reference, tracking specific aspect of a given item. Base item references are required by some dynamic items in Colorway, for example the dynamic table item.

**Parameters**

| | |
|---|---|
| *item* | The item we want to the reference to track. |
| *propertyName* | The name of the property of the item, the reference is to track. |

**Returns**

Newly created base item reference.

**See also**

addDynamicTableItem
Add a new dynamic table item

```
805 def createBaseItemReference( item, propertyName ):
806     from colorway.canvas import BaseItemReference
807     reference = BaseItemReference();
808     reference.call( 'setTrackedItem', item.owningDocument, item.identifier );
809     reference.trackedProperty = propertyName;
810     return reference;
811
```

#### 4.1.2.19 createDynamicSwatch()

```
def ColorwayPythonAPIExamples.createDynamicSwatch (
              swatchName,
              indexInPalette )
```

Create dynamic swatch from active palette.

Following example shows a simple function that creates a new dynamic color swatch. Dynamic swatch references and index in the active palette. If current palette is changed, the actual color of the swatch will be taken from the current palette using the index number.

**Parameters**

| | |
|---|---|
| *swatchName* | Name fo the newsly created swatch. |
| *indexInPalette* | The index number of the swatch in any palette. |

**Returns**

Newly created dynamic swatch.

**See also**

createStaticSwatch
setObjectSwatch

```
1332 def createDynamicSwatch( swatchName, indexInPalette ):
1333     from colorway import Swatch
1334     swatch = Swatch();
1335     swatch.stringIdentifier = swatchName;
1336     swatch.index = indexInPalette;
1337     return swatch;
1338
```

#### 4.1.2.20 createLink()

```
def ColorwayPythonAPIExamples.createLink (
              name,
              parentDirectory,
              linkTargetStringIdentifier )
```

Create a link in `parentDirectory` to a resource identified by `targetStringIdentifier`.

Following example shows a simple helper function that shows how to create a link to an asset in a given directory. In the Colorway API examples documented here, this function is used by addLinkAsItemVariant.

**Parameters**

| | |
|---|---|
| *name* | The name of the link. |
| *parentDirectory* | Handle to a live source directory, the link is to be added to. |
| *linkTargetStringIdentifier* | Path to the asset, the link is to point to. |

**Returns**

Newly created link.

**See also**

addLinkAsItemVariant
How to add new variants and select the active one

```
1010 def createLink( name, parentDirectory, linkTargetStringIdentifier ):
1011     from colorway            import Link;
1012     from colorway.livesource import Resource;
1013     asset = Link();
1014     asset.linkTarget = linkTargetStringIdentifier;
1015     resource         = Resource.createResource( parentDirectory, Resource.getResourceTypeById( 'link' ),
      asset, name + '.lnk', True, True );
1016     return resource;
1017
```

#### 4.1.2.21 createNewProject()

```
def ColorwayPythonAPIExamples.createNewProject (
              projectName,
              documentName )
```

Create new Colorway project.

Following example shows a simple wrapper function that can be used to create a new project.

Creating new project consists of three steps:

- creating an empty project,
- initializing the empty project,
- creating the main document, that will be the parent of all sheets.

**Warning**

A newly created project is empty. Before You can add any visible items, You need to create at least one sheet. Refer to addSheet to see how to add new sheet items to the project.
Newly created project is store in a temporary location, that will be erased if the project is closed, to prevent that make sure that You save the project to a permanent location. Please refer to saveProjectAs.

**Parameters**

| | |
|---|---|
| *projectName* | When creating a new Colorway project, You need to provide a name for it. |
| *documentName* | Before we can add any useful item to the project we need to create the main document and provide a name for it. |

**Returns**

To be able to add new elements to Colorway projects or manipulate it You need to keep around some certain objects. In this example the function returns a dictionary containing:

- project asset, under the key

  `'project'`

- resource handle of the project directory, under the key

  `'projectDirectoryHandle'`

- main document asset, under the key

  `'document'`

**See also**

openProject
saveProject
saveProjectAs
How to create new Colorway project

```
412 def createNewProject( projectName, documentName ):
413     from colorway.canvas import Project;
414     project = Project();
415     projectInfo = {};
416     projectInfo[ 'project'                ] = project;
417     projectInfo[ 'projectDirectoryHandle' ] = project.call( 'initializeProject', projectName, '' );
418     projectInfo[ 'document'               ] = project.call( 'createDocument', documentName );
419     return projectInfo;
420
```

#### 4.1.2.22 createStaticSwatch()

```
def ColorwayPythonAPIExamples.createStaticSwatch (
              swatchName,
              color )
```

Create new static swatch for a color.

Following example shows a simple function that creates a new static color swatch. Static swatch is independent and not a part of any palette, which means switching palettes, will not change the color assigned using this swatch.

**Parameters**

| | |
|---|---|
| *swatchName* | Name of the newly created swatch. |
| *color* | String identifying the new color in the hexadecimal code. For example: `'#ff0000'` for pure red color. |

**Returns**

Newly created static swatch.

**See also**

createDynamicSwatch
setObjectSwatch

```
1312 def createStaticSwatch( swatchName, color ):
1313     from colorway import Swatch
1314     swatch = Swatch();
1315     swatch.stringIdentifier = swatchName;
1316     swatch.color = color;
1317     return swatch;
1318
```

#### 4.1.2.23 exportVariationsAsIndividualPngs()

```
def ColorwayPythonAPIExamples.exportVariationsAsIndividualPngs (
              variations,
              document,
              outputDir,
              namePrefix )
```

Export list of variations as individual PNG files.

Following example shows a simple function that can be used to export specific variations as individual images in PNG format.

**Parameters**

| | |
|---|---|
| *variations* | List of variations to export. |
| *document* | Document to export. |
| *outputDir* | Path to the output directory. |
| *namePrefix* | Name added in front of each exported variation. |

**Warning**

The path defined by `outputDir` parameter must exist, otherwise the export will not be export will not be performed.

**See also**

exportVariationsAsPdf
exportVariationsAsPngLayers
exportVariationsAsPngSources
How to export Colorway project

```
523 def exportVariationsAsIndividualPngs( variations, document, outputDir, namePrefix ):
524     from colorway.canvas import Exporter
525     exporter = Exporter();
526     exporter.call( 'exportVariationsFlat', variations, document.dpi, outputDir, namePrefix, 'png', 100 );
527
```

#### 4.1.2.24 exportVariationsAsPdf()

```
def ColorwayPythonAPIExamples.exportVariationsAsPdf (
                variations,
                document,
                outputDir,
                namePrefix,
                asSingleFile )
```

Export list of variations in PDF format.

Following example shows a simple function that can be used to export specific variations as PDF document.

**Parameters**

| | |
|---|---|
| *variations* | List of variations to export. |
| *document* | Document to export. |
| *outputDir* | Path to the output directory. |
| *namePrefix* | Name added in front of each exported variation. |
| *asSingleFile* | Boolean flag indicating whether the variations should be exported as a single multi-page PDF document or as individual files. |

**Warning**

The path defined by `outputDir` parameter must exist, otherwise the export will not be export will not be performed.

**See also**

exportVariationsAsIndividualPngs
exportVariationsAsPngLayers
exportVariationsAsPngSources
How to export Colorway project

```
501 def exportVariationsAsPdf( variations, document, outputDir, namePrefix, asSingleFile ):
502     from colorway.canvas import Exporter
503     exporter = Exporter();
504     exporter.call( 'exportVariationsFlat', variations, document.dpi, outputDir, namePrefix, 'pdf', 100,
      asSingleFile );
505
```

#### 4.1.2.25 exportVariationsAsPngLayers()

```
def ColorwayPythonAPIExamples.exportVariationsAsPngLayers (
                variations,
                document,
                outputDir,
                namePrefix )
```

Export list of variations as individual layers in PNG format.

Following example shows a simple function that can be used to export specific variations as layers in PNG format. Each variation is split into layers, each layer containing a single Colorway item or asset. The layers are all of the same size allowing them to be composed in an image processing application.

**Parameters**

| | |
|---|---|
| *variations* | List of variations to export. |
| *document* | Document to export. |
| *outputDir* | Path to the output directory. |
| *namePrefix* | Name added in front of each exported variation. |

**Warning**

The path defined by `outputDir` parameter must exist, otherwise the export will not be export will not be performed.

**See also**

exportVariationsAsPdf
exportVariationsAsIndividualPngs
exportVariationsAsPngSources
How to export Colorway project

```
547 def exportVariationsAsPngLayers( variations, document, outputDir, namePrefix ):
548     from colorway.canvas import Exporter
549     exporter = Exporter();
550     exporter.call( 'exportVariationsLayered', variations, document.dpi, outputDir, namePrefix, 'png', 100 )
      ;
551
```

#### 4.1.2.26 exportVariationsAsPngSources()

```
def ColorwayPythonAPIExamples.exportVariationsAsPngSources (
                variations,
                document,
                outputDir,
                namePrefix )
```

Export sources for given variations as individual PNG files.

Following example shows a simple function that can be used to export specific variations as individual components in PNG format. Each variation is split into individual images containing single Colorway item or asset. The images are cropped to have minimal possible size.

**Parameters**

| | |
|---|---|
| *variations* | List of variations to export. |
| *document* | Document to export. |
| *outputDir* | Path to the output directory. |
| *namePrefix* | Name added in front of each exported variation. |

**Warning**

The path defined by `outputDir` parameter must exist, otherwise the export will not be export will not be performed.

**See also**

exportVariationsAsPdf
exportVariationsAsIndividualPngs
exportVariationsAsPngLayers
How to export Colorway project

```
570 def exportVariationsAsPngSources( variations, document, outputDir, namePrefix ):
571     from colorway.canvas import Exporter
572     exporter  = Exporter();
573     exporter.call( 'exportVariationsSources', variations, document.dpi, outputDir, namePrefix, 'png', 100 )
      ;
574
```

#### 4.1.2.27 getItemVariants()

```
def ColorwayPythonAPIExamples.getItemVariants (
              item )
```

Get a list of available variants.

Following example shows a simple function that can be used to get a list of all available variants (assets) in a give Colorway item.

**Parameters**

| | |
|---|---|
| *item* | Item, the variants of which we want to retrieve. |

**Returns**

A list containing resource handles for all available variants of the given item.

**See also**

How to add new variants and select the active one

```
1160 def getItemVariants( item ):
1161     resource = item.embeddedSourceEnsureExists.resource;
1162     resourceHandles = resource.getFilteredResources( item.filter );
1163     return resourceHandles;
1164
```

#### 4.1.2.28 getItemWorkingState()

```
def ColorwayPythonAPIExamples.getItemWorkingState (
              item,
              variation )
```

Get item's state for given item.

Following example shows a simple function that can be used to get the state of a given Colorway item, for a given variation.

**Parameters**

| | |
|---|---|
| *item* | Item the state of which we want to get. |
| *variation* | The variation of item's parent sheet we want the state to belong to. |

**Returns**

State of the given item in specific variation.

**See also**

How to add/remove variations

```
1242 def getItemWorkingState( item, variation ):
1243     return variation.call( 'getStateFor', item );
1244
```

#### 4.1.2.29 getPartNameToIdMap()

```
def ColorwayPythonAPIExamples.getPartNameToIdMap (
              multiPartItemState )
```

Create a dictionary mapping part's name to its numerical id.

Following example shows a simple function that can be used to create a dictionary, mapping part name to it's id number. Such a dictionary can be used to apply color swatches to multiple parts at once to a multi-part item. Please refer to applyPartNameToSwatchMapping to see an example of such application.

**Parameters**

| | |
|---|---|
| *multiPartItemState* | State of a multi-part item, i. e. either DCI item or vector image item. |

**Returns**

A dictionary having part names as keys, and part id numbers as values.

**See also**

applyPartNameToSwatchMapping
How to define new palettes and apply colors to items

```
1031 def getPartNameToIdMap( multiPartItemState ):
1032     ids = multiPartItemState.call( 'getAllLeafOrCollapsedPartIds' );
1033     map = {}
1034     for id in ids:
1035         part = multiPartItemState.call( 'getPartForId', id );
1036         map[ part.name] = id;
1037 
1038     return map;
1039 
```

#### 4.1.2.30 getResourcesOfType()

```
def ColorwayPythonAPIExamples.getResourcesOfType (
              directoryHandle,
              typeName )
```

Retrieve a list of all assets of specific type available in given project.

Following example shows a simple function that can be used to retrieve a list of all resources of specific type available in directory.

**Parameters**

| | |
|---|---|
| *directoryHandle* | Handle to a live source directory to search in. |
| *typeName* | The name of the type of asset to be retrieved. |

The `directoryHandle` for entire project can be obtain either when creating new project (see createNewProject) or when opening existing one (see openProject).

The `typeName` parameter should have one of the following values:

1. `'directory'`
   Use it to retrieve list of all directories available in given directory/project.
2. `'document'`
   Use it to retrieve list of all documents available in given directory/project.
3. `'dci'`
   Use it to retrieve list of all DCI items available in given directory/project.
4. `'image'`
   Use it to retrieve list of all image items available in given directory/project.
5. `'link'`
   Use it to retrieve list of all links available in given directory/project.
6. `'material'`
   Use it to retrieve list of all materials available in given directory/project.
7. `'materialPalette'`
   Use it to retrieve list of all material palettes available in given directory/project.

8. `'palette'`

   Use it to retrieve list of all palettes available in given directory/project.

9. `'repository'`

   Use it to retrieve list of all repositories available in given directory/project.

10. `'swatch'`

    Use it to retrieve list of all swatches available in given directory/project.

11. `'table'`

    Use it to retrieve list of all table items available in given directory/project.

12. `'text'`

    Use it to retrieve list of all text item available in given directory/project.

**Returns**

List containing resource handles of specified types of objects contained in a given directory.

**See also**

createNewProject
openProject
How to find and manipulate existing item in a project

```
1137 def getResourcesOfType( directoryHandle, typeName ):
1138     from colorway.livesource import Resource;
1139     from colorway.livesource import ResourceFilter;
1140 
1141     directoryResource = directoryHandle.resource;
1142 
1143     filter = ResourceFilter();
1144     filter.resourceTypes = [ Resource.getResourceTypeById( typeName ) ];
1145 
1146     resourceHandles = directoryResource.getFilteredResources( filter );
1147 
1148     return resourceHandles;
1149 
```

#### 4.1.2.31 openProject()

```
def ColorwayPythonAPIExamples.openProject (
              path )
```

Open existing Colorway project.

Following example shows a simple function that can be used to open existing project in a given location.

**Parameters**

| | |
|---|---|
| *path* | Location of the project to be opened prefixed with "/file/". For example, if the location to be opened is: "/home/user1/CWProject.cway", the `path` parameter should be set to "/file//home/user1/CWProject.cway". |

**Returns**

A dictionary containing:

- the asset of the opened project, under the key:

  `'project'`

- the resource handle of the project directory, under the key:

  `'projectDirectoryHandle'`

- an error message string containing the cause of an error if any occurs during opening of the project.

**See also**

createProject
saveProject
saveProjectAs
How to open existing Colorway project

```
439 def openProject( path ):
440     result = colorway.canvas.Project.openProject( path );
441     project = result[ 'project' ];
442     documentsDictoryHandle = project.documentsDirectory
443     documentsHandles       = getResourcesOfType( documentsDictoryHandle, 'document' );
444     result[ 'document' ]   = documentsHandles[0].finalAsset;
445     return result;
446
```

#### 4.1.2.32 removeVariation()

```
def ColorwayPythonAPIExamples.removeVariation (
              variation,
              sheet )
```

Remove given variation form given sheet.

Following example shows a simple function that can be used to remove specific variation from a specific sheet.

**Parameters**

| | |
|---|---|
| *variation* | Variation to be removed. |
| *sheet* | Sheet the variation to be removed is stored in. |

**See also**

addVariation
addSheet
Add new variation to the given sheet

```
925 def removeVariation( variation, sheet ):
926     sheet.call( 'removeVariation', variation );
927
```

#### 4.1.2.33 saveProject()

```
def ColorwayPythonAPIExamples.saveProject (
              project )
```

Save changes to a Colorway project.

The following example shows a simple function that can be used to save changes to a project.

**Parameters**

| | |
|---|---|
| *project* | Asset of the project to be saved. |

**Warning**

A newly created project is by default stored in a temporary location that is going to be erased when the project is closed. Simply saving changes the project, will not prevent the project from being erased upon closing. Please refer to saveProjectAs to see how to specify a more permanent location for a new project.

**See also**

saveProjectAs
createNewProject
openProject
How to save a Colorway project

```
462 def saveProject( project ):
463     projectDir = project.projectDirectory.resource;
464     projectDir.call( 'save', False, 'SaveMode_Children' );
465
```

#### 4.1.2.34 saveProjectAs()

```
def ColorwayPythonAPIExamples.saveProjectAs (
              project,
              path )
```

Save Colorway project to location.

Following example shows a simple function that can be used to save Colorway project to specific location.

**Parameters**

| | |
|---|---|
| *project* | Handle to the project to be save. |
| *path* | The location the project is to be saved to. |

**See also**

saveProject
createNewProject
openProject
How to save a Colorway project

```
478 def saveProjectAs( project, path ):
479     projectDir = project.projectDirectory.resource;
480     projectDir.call( 'saveAs', path, False, 'SaveMode_Children' );
481
```

#### 4.1.2.35 selectActivePalette()

```
def ColorwayPythonAPIExamples.selectActivePalette (
              sheet,
              paletteH )
```

Select active palette for a sheet.

Following examples shows a simple function that selects given palette as an active one for a given sheet.

**Parameters**

| | |
|---|---|
| *sheet* | Sheet, active palette is to be changed for. |
| *paletteH* | Palette handle for the palette to be selected as active. |

**See also**

addPalette
How to define new palettes and apply colors to items

```
1349 def selectActivePalette( sheet, paletteH ):
1350     sheet.workingVariation.selectedPalette = paletteH;
1351
```

#### 4.1.2.36 setFontItemStyle()

```
def ColorwayPythonAPIExamples.setFontItemStyle (
              fontItem,
              fontFamily,
              fontPointSize,
              isBold,
              isItalic,
              isUnderline )
```

Set the style of an item that uses a font.

A number of colorway items such as: text items, palette items, contact sheet items and table items, display text as part of their visual representation.

Colorway item's that can display text are:

- text items (see section: Add a new text item),
- palette items (see section: Add a new palette item),
- contact sheet items (see section: Add a new contact sheet item),
- table items (see section: Add a new dynamic table item).

Following example shows a simple function that can be used to set the most common properties of a font.

**Parameters**

| | |
|---|---|
| *fontItem* | The item |
| *fontFamily* | String containing the name of the font's family. |
| *fontPointSize* | The point size of the font. |
| *isBold* | Boolean flag indicating if the font should use the bold style. |
| *isItalic* | Boolean flag indicating if the font should use the italic style. |
| *isUnderline* | Boolean flag indicating if the font should use the underline style. |

**Warning**

Not every font supports all style variants. If a font does not support requested style, Colorway will attempt to select the best matching style available in selected font.

**See also**

Change the font family, font style and font size of Colorway item's with text.

```
975 def setFontItemStyle( fontItem, fontFamily, fontPointSize, isBold, isItalic, isUnderline ):
976     fontItem.textStyle.fontFamily    = fontFamily;
977     fontItem.textStyle.fontPointSize = fontPointSize;
978     fontItem.textStyle.fontBold      = isBold;
979     fontItem.textStyle.fontItalic    = isItalic;
980     fontItem.textStyle.fontUnderline = isUnderline;
981
```

#### 4.1.2.37 setItemPosition()

```
def ColorwayPythonAPIExamples.setItemPosition (
              item,
              x,
              y )
```

Set the global position of a given item: Following example shows a simple function that can be used to set the global position of an item.

**Parameters**

| | |
|---|---|
| *item* | Item, the position of which we want to set. |
| *x* | New x-axis position in pixels. |
| *y* | New y-axis position in pixels. |

```
1253 def setItemPosition( item, x, y ):
1254     state = item.owningSheet.workingVariation.call( 'getStateFor', item );
1255     state.worldTransform.position = ( x, y );
1256
```

#### 4.1.2.38 setItemRotation()

```
def ColorwayPythonAPIExamples.setItemRotation (
              item,
              angle )
```

Set the rotation of a given item.

The following example shows a simple function that can be used to set the rotation of an item.

**Parameters**

| | |
|---|---|
| *item* | Item, the rotation of which we want to set. |
| *angle* | Target rotation of the given item in degrees. |

```
1275 def setItemRotation( item, angle ):
1276     state = item.owningSheet.workingVariation.call( 'getStateFor', item);
1277     state.worldTransform.rotation = angle;
1278
```

#### 4.1.2.39 setItemSize()

```
def ColorwayPythonAPIExamples.setItemSize (
              item,
              width,
              height )
```

Set the size of a given item: The following example shows a simple function that can be used to set the size of an item.

**Parameters**

| | |
|---|---|
| *item* | Item, the size of which we want to set. |
| *width* | New width in pixels. |
| *height* | New height in pixels. |

```
1264 def setItemSize( item, width, height ):
1265     state = item.owningSheet.workingVariation.call( 'getStateFor', item);
1266     state.size = ( width, height );
1267
```

#### 4.1.2.40 setObjectSwatch()

```
def ColorwayPythonAPIExamples.setObjectSwatch (
              objectWithColor,
              sheet,
              assignmentPoint,
              swatch )
```

Assign color to an object.

The following example shows a simple function that can be used to assing a color an object.

**Parameters**

| | |
|---|---|
| *objectWithColor* | Object, the color of which we want to set. |
| *sheet* | Parent sheet, the given object is placed on. |
| *assignmentPoint* | Number identifying what the color should be applied to, please refer to ColorAssignmentPoint for more information. |
| *swatch* | Swatch with the color to be assigned. To see how to create swatches refer to ColorwayPythonAPIExamples.createSwatch. |

**See also**

ColorAssignmentPoint
createStaticSwatch
createDynamicSwatch

```
1294 def setObjectSwatch( objectWithColor, sheet, assignmentPoint, swatch ):
1295     colorAssignment = sheet.workingVariation.colorAssignment;
1296     colorAssignment.call( 'setObjectSwatch', objectWithColor, assignmentPoint, swatch );
1297
```

#### 4.1.2.41 setResourceHandleAsActiveVariant()

```
def ColorwayPythonAPIExamples.setResourceHandleAsActiveVariant (
              item,
              resourceHandle,
              variation )
```

Set resource handle as active variant of an item.

Following example shows a simple function that shows how to select the active variant within an item.

**Parameters**

| | |
|---|---|
| *item* | The item we want to select the active variant for. |
| *resourceHandle* | Resource handle of the variant, we want to become active one. |
| *variation* | Variation, in which the specific resource handle is to be selected as active one. |

**See also**

addLinkAsItemVariant
How to add new variants and select the active one

```
992 def setResourceHandleAsActiveVariant( item, resourceHandle, variation ):
993     if resourceHandle.call( 'isChildOf', item.liveSource ):
994         itemState                = variation.call( 'getStateFor', item );
995         itemState.source.variant = resourceHandle;
996
```

#### 4.1.2.42 setWorldTransform()

```
def ColorwayPythonAPIExamples.setWorldTransform (
              itemState,
              positionX,
              positionY,
              pivotX,
              pivotY,
              rotation,
              scaleX,
              scaleY )
```

Set the world position, rotation and scale for a given item.

The following example shows a simple functino that can be used to transform an item, i.e. set it's position, rotation and scale.

**Parameters**

| | |
|---|---|
| *itemState* | State of the item to be transformed. |
| *positionX* | The x-axis position on canvas. In pixels. |
| *positionY* | The y-axis position on canvas. In pixels. |
| *pivotX* | The x-axis position of the transformation pivot. In pixels. |
| *pivotY* | The y-axis position of the transformation pivot. In pixels. |
| *rotation* | Final rotation angle in degrees. |
| *scaleX* | The relative x-axis scale of the item. |
| *scaleY* | The relative y-axis scale of the item. |

**See also**

setItemPosition
setItemSize
setItemRotation
How to set the position, size and rotation of a Colorway item.

```
1370 def setWorldTransform( itemState, positionX, positionY, pivotX, pivotY, rotation, scaleX, scaleY ):
1371     itemState.worldTransform.position = ( positionX, positionY );
1372     itemState.worldTransform.pivot    = ( pivotX,    pivotY    );
1373     itemState.worldTransform.rotation = rotation;
1374     itemState.worldTransform.scale    = ( scaleX,    scaleY    );
1375
```

### 4.1.3 Variable Documentation

#### 4.1.3.1 TypesToConsolidate

```
list ColorwayPythonAPIExamples.TypesToConsolidate
```

**Initial value:**

```
1 =  [
2     Resource.getResourceTypeById( 'dci' ),
3     Resource.getResourceTypeById( 'image' ),
4     Resource.getResourceTypeById( 'palette' ),
5     Resource.getResourceTypeById( 'table' ),
6     Resource.getResourceTypeById( 'text' ),
7 ];
```

List of types to include when consolidating a project.

Following example constructs a simple list of type identifiers, for types that should be included when consolidating a project. You can simply pass it to consolidateProject when consolidating a project.

**Warning**

To use the `Resource` type, You must import the `colorway.livesource.Resource` module using following code:

```
from colorway.livesource import Resource
```

**See also**

consolidateProject
How to consolidate Colorway project

# Chapter 5

# Class Documentation

## 5.1 ColorwayPythonAPIExamples.ColorAssignmentPoint

Allowed values for colorAssignmentPoint parameter.

### Static Public Attributes

- int BackColor = 3;

  *Assign color as back color.*
- int BorderColor = 2;

  *Assign color as border color.*
- int FontColor = 1;

  *Assign color as font color.*
- int PrimaryColor = -1;

  *Assign color as primary color.*
- int SecondaryBackColor = 4;

  *Assign color as secondary back color.*
- int SpecularColor = 1;

  *Assign color as specular color.*
- int StrokeColor = 2;

  *Assign color as stroke color.*
- int TintColor = 0;

  *Assign color as tint color.*

### 5.1.1 Detailed Description

Allowed values for colorAssignmentPoint parameter.

Following simple class stores the allowed values of the color assignment point parameter used by: setObjectSwatch

**See also**

setObjectSwatch
How to define new palettes and apply colors to items

### 5.1.2 Member Data Documentation

#### 5.1.2.1 BackColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.BackColor = 3;  [static]
```

Assign color as back color.

#### 5.1.2.2 BorderColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.BorderColor = 2;  [static]
```

Assign color as border color.

#### 5.1.2.3 FontColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.FontColor = 1;  [static]
```

Assign color as font color.

#### 5.1.2.4 PrimaryColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.PrimaryColor = -1;  [static]
```

Assign color as primary color.

#### 5.1.2.5 SecondaryBackColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.SecondaryBackColor = 4;  [static]
```

Assign color as secondary back color.

#### 5.1.2.6 SpecularColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.SpecularColor = 1;  [static]
```

Assign color as specular color.

#### 5.1.2.7 StrokeColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.StrokeColor = 2;  [static]
```

Assign color as stroke color.

#### 5.1.2.8 TintColor

```
int ColorwayPythonAPIExamples.ColorAssignmentPoint.TintColor = 0;  [static]
```

Assign color as tint color.

# Index