
OpenColorIO Configuration for ACES Documentation

Release 0.1.0

OpenColorIO Contributors

Aug 09, 2022

CONTENTS

1	1.1	Features	3
2	1.2	Installation	5
2.1	1.2.1	Docker	5
2.2	1.2.2	Pypi	5
	2.2.1	1.2.2.1 Primary Dependencies	5
	2.2.2	1.2.2.2 Plotting Dependencies	5
	2.2.3	1.2.2.3 Development Dependencies	6
3	1.3	Usage	7
3.1	1.3.1	Tasks	7
3.2	1.3.2	API	7
	3.2.1	OpenColorIO Configuration for ACES - Manual	7
4	1.4	About	25
		Index	27

WARNING: This repository is under construction!

The [OpenColorIO Configuration for ACES](#) is an open-source [Python](#) package implementing support for the generation of the *OCIO* configurations for the [Academy Color Encoding System](#) (ACES).

It is freely available under the [New BSD License](#) terms.

Table of Contents

- 1 *OpenColorIO Configuration for ACES*
 - 1.1 *Features*
 - 1.2 *Installation*
 - * 1.2.1 *Docker*
 - * 1.2.2 *Pypi*
 - 1.3 *Usage*
 - * 1.3.1 *Tasks*
 - * 1.3.2 *API*
 - 1.4 *About*

1.1 FEATURES

The following features are available:

- Automatic *OCIO Reference* configuration generation for *aces-dev CTL* reference implementation.
- Configurable generator producing the *OCIO Studio* configuration.

1.2 INSTALLATION

2.1 1.2.1 Docker

Installing the dependencies for the [previous config generator](#) was not a trivial task. For ease of use an [aswf-docker](#) based container is now available.

Creating the container from the [Dockerfile](#) is done as follows:

```
docker build -t aswf/opencolorio-config-aces:latest .
```

or alternatively, if the dependencies described in the next section are satisfied:

```
invoke docker build
```

Then, to run *bash* in the container:

```
docker run -it -v ${PWD}:/home/aswf/OpenColorIO-Config-ACES aswf/opencolorio-config-aces:latest /  
↪ bin/bash
```

2.2 1.2.2 Pypi

The **OpenColorIO Configuration for ACES** package requires various dependencies in order to run and be able to generate the *OCIO* configurations:

2.2.1 1.2.2.1 Primary Dependencies

- `python>=3.7`
- `networkx`
- `OpenColorIO`

2.2.2 1.2.2.2 Plotting Dependencies

- `graphviz`
- `pygraphviz`

2.2.3 1.2.2.3 Development Dependencies

- coverage
- coveralls
- flake8
- invoke
- nose
- pre-commit
- pytest
- restructuredtext-lint
- sphinx
- sphinx-rtd-theme
- twine
- yapf=0.23.0

Once the dependencies are satisfied, the **OpenColorIO Configuration for ACES** package can be installed from the [Python Package Index](#) by issuing this command in a shell:

```
pip install --user opencolorio-config-aces
```

1.3 USAGE

3.1 1.3.1 Tasks

Various tasks are currently exposed via `invoke`.

This is currently the recommended way to build the configuration until a dedicated CLI is provided.

Listing the tasks is done as follows:

```
invoke --list
```

Assuming the dependencies are satisfied, the task to build the reference configuration is:

```
invoke build-reference-config
```

Alternatively, with the docker container built:

```
invoke docker-run-build-reference-config
```

3.2 1.3.2 API

The main reference for `OpenColorIO Configuration for ACES` is the `manual`.

3.2.1 OpenColorIO Configuration for ACES - Manual

Reference

`OpenColorIO Configuration for ACES`

Generation

- *Config Generation Common Objects*
- *Reference Configuration*
 - *aces-dev Discovery*
 - *aces-dev Conversion Graph*
 - *aces-dev Reference Config Generator*

Config Generation Common Objects

opencolorio_config_aces

<code>colorspace_factory(name[, family, encoding, ...])</code>	<i>OpenColorIO</i> colorspace factory.
<code>view_transform_factory(name[, family, ...])</code>	<i>OpenColorIO</i> view transform factory.
<code>ConfigData(profile_version, description, ...)</code>	Defines the data container for an <i>OpenColorIO</i> config.
<code>validate_config(config)</code>	Validates given <i>OpenColorIO</i> config.
<code>generate_config(data[, config_name, validate])</code>	Generates the <i>OpenColorIO</i> config from given data.

opencolorio_config_aces.colorspace_factory

`opencolorio_config_aces.colorspace_factory(name, family=None, encoding=None, categories=None, description=None, equality_group=None, bit_depth=None, allocation=None, allocation_vars=None, to_reference=None, from_reference=None, is_data=None, reference_space=None, base_colorspace=None)`

OpenColorIO colorspace factory.

Parameters

- **name** (unicode) – *OpenColorIO* colorspace name.
- **family** (unicode, optional) – *OpenColorIO* colorspace family.
- **encoding** (unicode, optional) – *OpenColorIO* colorspace encoding.
- **categories** (unicode or array_like, optional) – *OpenColorIO* colorspace categories.
- **description** (unicode, optional) – *OpenColorIO* colorspace description.
- **equality_group** (unicode, optional) – *OpenColorIO* colorspace equality_group.
- **bit_depth** (int, optional) – *OpenColorIO* colorspace bit depth.
- **allocation** (int, optional) – *OpenColorIO* colorspace allocation type.
- **allocation_vars** (tuple, optional) – *OpenColorIO* colorspace allocation variables.
- **to_reference** (object, optional) – *To Reference OpenColorIO* colorspace transform.
- **from_reference** (object, optional) – *From Reference OpenColorIO* colorspace transform.
- **reference_space** (ReferenceSpaceType, optional) – *OpenColorIO* colorspace reference space.
- **is_data** (bool, optional) – Whether the colorspace represents data.
- **base_colorspace** (ColorSpace, optional) – *OpenColorIO* base colorspace inherited for bit depth, allocation, allocation variables, and to/from reference transforms.

Returns *OpenColorIO* colorspace.

Return type ColorSpace

opencolorio_config_aces.view_transform_factory

```
opencolorio_config_aces.view_transform_factory(name, family=None, categories=None, description=None,
                                              to_reference=None, from_reference=None,
                                              reference_space=None,
                                              base_view_transform=None)
```

OpenColorIO view transform factory.

Parameters

- **name** (unicode) – *OpenColorIO* view transform name.
- **family** (unicode, optional) – *OpenColorIO* view transform family.
- **categories** (array_like, optional) – *OpenColorIO* view transform categories.
- **description** (unicode, optional) – *OpenColorIO* view transform description.
- **to_reference** (object, optional) – To Reference *OpenColorIO* view transform transform.
- **from_reference** (object, optional) – From Reference *OpenColorIO* view transform transform.
- **reference_space** (ReferenceSpaceType, optional) – *OpenColorIO* view transform reference space.
- **base_view_transform** (ViewTransform, optional) – Inherited *OpenColorIO* base view transform.

Returns *OpenColorIO* view transform.

Return type ViewTransform

opencolorio_config_aces.ConfigData

```
class opencolorio_config_aces.ConfigData(profile_version: int = 1, description: str = 'An "Open-ColorIO" config generated by "OpenColorIO-Config-ACES".', roles: Union[dict, collections.OrderedDict] = <factory>, colorspace: Union[list, tuple] = <factory>, looks: Union[list, tuple] = <factory>, view_transforms: Union[list, tuple] = <factory>, shared_views: Union[list, tuple] = <factory>, views: Union[list, tuple] = <factory>, active_displays: Union[list, tuple] = <factory>, active_views: Union[list, tuple] = <factory>, file_rules: Union[list, tuple] = <factory>, viewing_rules: Union[list, tuple] = <factory>, inactive_colorspaces: Union[list, tuple] = <factory>, default_view_transform: str = <factory>)
```

Defines the data container for an *OpenColorIO* config.

Parameters

- **profile_version** (int, optional) – Config major version, i.e. 1 or 2.
- **description** (unicode, optional) – Config description.
- **roles** (dict) – Config roles, a dict of role and colorspace name.
- **colorspaces** (array_like) – Config colorspace, an iterable of PyOpenColorIO. ColorSpace class instances.

- **looks** (array_like, optional) – Config looks, an iterable of PyOpenColorIO. Look class instances.
- **view_transforms** (array_like, optional) – Config view transforms, an iterable of PyOpenColorIO.ViewTransform class instances.
- **shared_views** (array_like, optional) – Config shared views, an iterable of dicts of view, view transform, colorspace and rule names, iterable of looks and description.
- **views** (array_like, optional) – Config views, an iterable of dicts of display, view and colorspace names.
- **active_displays** (array_like, optional) – Config active displays, an iterable of display names.
- **active_views** (array_like, optional) – Config active displays, an iterable of view names.
- **file_rules** (array_like, optional) – Config file rules, a dict of file rules.
- **viewing_rules** (array_like, optional) – Config viewing rules, a dict of viewing rules.
- **inactive_colorspaces** (array_like, optional) – Config inactive colorspaces an iterable of colorspace names.
- **default_view_transform** (unicode, optional) – Name of the default view transform.

profile_version

Type `int`

description

Type `str`

roles

Type `Union[dict, collections.OrderedDict]`

colorspaces

Type `Union[list, tuple]`

looks

Type `Union[list, tuple]`

view_transforms

Type `Union[list, tuple]`

shared_views

Type `Union[list, tuple]`

views

Type `Union[list, tuple]`

active_displays

Type `Union[list, tuple]`

active_views

Type `Union[list, tuple]`

file_rules

Type `Union[list, tuple]`

viewing_rules

Type Union[list, tuple]

inactive_colorspaces

Type Union[list, tuple]

default_view_transform

Type str

__init__(profile_version: int = 1, description: str = 'An "OpenColorIO" config generated by "OpenColorIO-Config-ACES".', roles: Union[dict, collections.OrderedDict] = <factory>, colorspaces: Union[list, tuple] = <factory>, looks: Union[list, tuple] = <factory>, view_transforms: Union[list, tuple] = <factory>, shared_views: Union[list, tuple] = <factory>, views: Union[list, tuple] = <factory>, active_displays: Union[list, tuple] = <factory>, active_views: Union[list, tuple] = <factory>, file_rules: Union[list, tuple] = <factory>, viewing_rules: Union[list, tuple] = <factory>, inactive_colorspaces: Union[list, tuple] = <factory>, default_view_transform: str = <factory>) → None

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__ ([profile_version, description, ...])	Initialize self.
---	------------------

Attributes

description

profile_version

opencolorio_config_aces.validate_config

opencolorio_config_aces.**validate_config**(config)

Validates given *OpenColorIO* config.

Parameters **config** (Config) – *OpenColorIO* config to validate.

Returns Whether the *OpenColorIO* config is valid.

Return type bool

opencolorio_config_aces.generate_config

opencolorio_config_aces.**generate_config**(data, config_name=None, validate=True)

Generates the *OpenColorIO* config from given data.

Parameters

- **data** (ConfigData) – *OpenColorIO* config data.
- **config_name** (unicode, optional) – *OpenColorIO* config file name, if given the config will be written to disk.
- **validate** (bool, optional) – Whether to validate the config.

Returns *OpenColorIO* config.

Return type Config

Reference Configuration

aces-dev Discovery

opencolorio_config_aces

<code>discover_aces_ctl_transforms([root_directory])</code>	Discovers the <i>ACES CTL</i> transform paths in given root directory: The given directory is traversed and *.ctl files are collected.
<code>classify_aces_ctl_transforms(...)</code>	Classifies given <i>ACES CTL</i> transforms.
<code>unclassify_ctl_transforms(...)</code>	Unclassifies given <i>ACES CTL</i> transforms.
<code>filter_ctl_transforms(ctl_transforms[, ...])</code>	Filters given <i>ACES CTL</i> transforms with given filters.
<code>print_aces_taxonomy()</code>	Prints <i>aces-dev</i> taxonomy:

opencolorio_config_aces.discover_aces_ctl_transforms

```
opencolorio_config_aces.discover_aces_ctl_transforms(root_directory='/home/docs/checkouts/readthedocs.org/user_
config-aces/envs/v0.1.0/lib/python3.7/site-
packages/opencolorio_config_aces/config/reference/aces-
dev/transforms/ctl')
```

Discovers the *ACES CTL* transform paths in given root directory: The given directory is traversed and *.ctl files are collected.

Parameters `root_directory` (unicode) – Root directory to traverse to find the *ACES CTL* transforms.

Returns

$$\{ \textit{directory}'_1 : [\textit{transform}_a.\textit{ctl}', \textit{transform}_b.\textit{ctl}'], \dots, \textit{directory}'_n : [\textit{transform}_c.\textit{ctl}', \textit{transform}_d.\textit{ctl}'] \}$$
Return type `dict`

Examples

```
>>> ctl_transforms = discover_aces_ctl_transforms()
>>> key = sorted(ctl_transforms.keys())[0]
>>> os.path.basename(key)
'ACEScc'
>>> sorted([os.path.basename(path) for path in ctl_transforms[key]])
['ACEScsc.Academy.ACES_to_ACEScc.ctl', 'ACEScsc.Academy.ACEScc_to_ACES.ctl']
```

opencolorio_config_aces.classify_aces_ctl_transforms

opencolorio_config_aces.classify_aces_ctl_transforms(*unclassified_ctl_transforms*)

Classifies given *ACES CTL* transforms.

Parameters *unclassified_ctl_transforms* (*dict*) – Unclassified *ACES CTL* transforms as returned by `opencolorio_config_aces.discover_aces_ctl_transforms()` definition.

Returns

$$\{“family”_1 : \{“genus”_1 : \{ \}_{CTL_1}, \dots, “family”_n : \{“genus”_2 : \{ \}_{CTL_2} \} \}$$

where

$$\{ \}_{CTL_n} = \{ “basename”_n : CTLTransform_n, \dots, “basename”_{n+1} : CTLTransform_{n+1} \}$$

Return type *dict*

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(
...     discover_aces_ctl_transforms())
>>> family = sorted(ctl_transforms.keys())[0]
>>> str(family)
'csc'
>>> genera = sorted(ctl_transforms[family])
>>> print(genera)
['ACEScc', 'ACEScct', 'ACEScg', 'ACESproxy', 'ADX', 'arri', 'canon', 'panasonic', 'red', 'sony
↪']
>>> genus = genera[0]
>>> sorted(ctl_transforms[family][genus].items())
[(‘ACEScsc.Academy.ACEScc’, CTLTransformPair(CTLTransform(‘csc...ACEScc...ACEScsc.Academy.ACES_
↪to_ACEScc.ctl’), CTLTransform(‘csc...ACEScc...ACEScsc.Academy.ACES_to_ACEScc.ctl’)))]
```

opencolorio_config_aces.unclassify_ctl_transforms

opencolorio_config_aces.unclassify_ctl_transforms(*classified_ctl_transforms*)

Unclassifies given *ACES CTL* transforms.

Parameters *classified_ctl_transforms* (*dict*) – Classified *ACES CTL* transforms as returned by `opencolorio_config_aces.classify_aces_ctl_transforms()` definition.

Returns

$$[CTLTransform_1, \dots, CTLTransform_n]$$

Return type *list*

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(
...     discover_aces_ctl_transforms())
>>> sorted(
...     unclassify_ctl_transforms(ctl_transforms), key=lambda x: x.path)[0]
CTLTransform('csc...ACEScc...ACEScsc.Academy.ACES_to_ACEScc.ctl')
```

opencolorio_config_aces.filter_ctl_transforms

opencolorio_config_aces.**filter_ctl_transforms**(ctl_transforms, filterers=None)
Filters given *ACES CTL* transforms with given filterers.

Parameters

- **ctl_transforms** (dict or list) – *ACES CTL* transforms as returned by `opencolorio_config_aces.classify_aces_ctl_transforms()` or `opencolorio_config_aces.unclassify_aces_ctl_transforms()` definitions.
- **filterers** (array_like, optional) – List of callables used to filter the *ACES CTL* transforms, each callable takes an *ACES CTL* transform as argument and returns whether to include or exclude the *ACES CTL* transform as a bool.

Returns

$[CTLTransform_1, \dots, CTLTransform_n]$

Return type `list`

Warning:

- This definition will forcibly unclassify the given *ACES CTL* transforms and return a flattened list.

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(
...     discover_aces_ctl_transforms())
>>> sorted(
...     filter_ctl_transforms(ctl_transforms, [lambda x: x.genus == 'p3']),
...     key=lambda x: x.path)[0]
CTLTransform('odt...p3...InvODT.Academy.P3D60_48nits.ctl')
```

opencolorio_config_aces.print_aces_taxonomy

opencolorio_config_aces.**print_aces_taxonomy**()
Prints *aces-dev* taxonomy:

- The *aces-dev CTL* transforms are discovered by traversing the directory defined by the `opencolorio_config_aces.config.reference.ACES_CTL_TRANSFORMS_ROOT` attribute using the `opencolorio_config_aces.discover_aces_ctl_transforms()` definition.
- The *CTL* transforms are classified by family e.g. *output_transform*, and genus e.g. *dcdm* using the `opencolorio_config_aces.classify_aces_ctl_transforms()` definition.
- The resulting datastructure is printed.

aces-dev Conversion Graph

opencolorio_config_aces

<code>build_aces_conversion_graph(ctl_transforms)</code>	Builds the <i>aces-dev</i> conversion graph from given <i>ACES CTL</i> transforms.
<code>node_to_ctl_transform(graph, node)</code>	Returns the <i>ACES CTL</i> transform from given node name.
<code>ctl_transform_to_node(graph, ctl_transform)</code>	Returns the node name from given <i>ACES CTL</i> transform.
<code>filter_nodes(graph[, filterers])</code>	Filters given <i>aces-dev</i> conversion graph nodes with given filterers.
<code>conversion_path(graph, source, target)</code>	Returns the conversion path from the source node to the target node in the <i>aces-dev</i> conversion graph.
<code>plot_aces_conversion_graph(graph, filename)</code>	Plots given <i>aces-dev</i> conversion graph using <i>Graphviz</i> and <i>pygraphviz</i> .

opencolorio_config_aces.build_aces_conversion_graph

`opencolorio_config_aces.build_aces_conversion_graph(ctl_transforms)`
 Builds the *aces-dev* conversion graph from given *ACES CTL* transforms.

Parameters `ctl_transforms` (`dict` or `list`) – *ACES CTL* transforms as returned by `opencolorio_config_aces.classify_aces_ctl_transforms()`, `opencolorio_config_aces.unclassify_aces_ctl_transforms()` or `opencolorio_config_aces.filter_aces_ctl_transforms()` definitions.

Returns *aces-dev* conversion graph.

Return type `DiGraph`

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(
...     discover_aces_ctl_transforms())
>>> build_aces_conversion_graph(ctl_transforms)
<networkx.classes.digraph.DiGraph object at 0x...>
```

opencolorio_config_aces.node_to_ctl_transform

`opencolorio_config_aces.node_to_ctl_transform(graph, node)`
 Returns the *ACES CTL* transform from given node name.

Parameters

- **graph** (`DiGraph`) – *aces-dev* conversion graph.
- **node** (`unicode`) – Node name to return the *ACES CTL* transform from.

Returns *ACES CTL* transform.

Return type `CTLTransform`

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(  
...     discover_aces_ctl_transforms())  
>>> graph = build_aces_conversion_graph(ctl_transforms)  
>>> node_to_ctl_transform(graph, 'ODT/P3D60_48nits')  
CTLTransform('odt...p3...ODT.Academy.P3D60_48nits.ctl')
```

opencolorio_config_aces.ctl_transform_to_node

opencolorio_config_aces.**ctl_transform_to_node**(graph, ctl_transform)

Returns the node name from given *ACES CTL* transform.

Parameters

- **graph** (DiGraph) – *aces-dev* conversion graph.
- **ctl_transform** (CTLTransform) – *ACES CTL* transform to return the node name from.

Returns Node name.

Return type unicode

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(  
...     discover_aces_ctl_transforms())  
>>> graph = build_aces_conversion_graph(ctl_transforms)  
>>> ctl_transform = node_to_ctl_transform(graph, 'ODT/P3D60_48nits')  
>>> ctl_transform_to_node(graph, ctl_transform)  
'ODT/P3D60_48nits'
```

opencolorio_config_aces.filter_nodes

opencolorio_config_aces.**filter_nodes**(graph, filterers=None)

Filters given *aces-dev* conversion graph nodes with given filterers.

Parameters

- **graph** (DiGraph) – *aces-dev* conversion graph.
- **filterers** (array_like, optional) – List of callables used to filter the *ACES CTL* transforms, each callable takes an *ACES CTL* transform as argument and returns whether to include or exclude the *ACES CTL* transform as a bool.

Returns Filtered *aces-dev* conversion graph nodes.

Return type list

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(
...     discover_aces_ctl_transforms())
>>> graph = build_aces_conversion_graph(ctl_transforms)
>>> sorted(filter_nodes(graph, [lambda x: x.genus == 'p3']))[0]
'InvRRTODT/P3D65_1000nits_15nits_ST2084'
```

opencolorio_config_aces.conversion_path

opencolorio_config_aces.**conversion_path**(*graph*, *source*, *target*)

Returns the conversion path from the source node to the target node in the *aces-dev* conversion graph.

Parameters

- **graph** (DiGraph) – *aces-dev* conversion graph.
- **source** (unicode) – Source node.
- **target** (unicode) – Target node.

Returns Conversion path from the source node to the target node.

Return type [list](#)

Examples

```
>>> ctl_transforms = classify_aces_ctl_transforms(
...     discover_aces_ctl_transforms())
>>> graph = build_aces_conversion_graph(ctl_transforms)
>>> conversion_path(graph, 'IDT/Venice_SLog3_SGamut3', 'ODT/P3D60_48nits')
[('IDT/Venice_SLog3_SGamut3', 'ACES2065-1'), ('ACES2065-1', 'OCES'), ('OCES', 'ODT/P3D60_48nits
↪ ')]
```

opencolorio_config_aces.plot_aces_conversion_graph

opencolorio_config_aces.**plot_aces_conversion_graph**(*graph*, *filename*, *prog*='dot', *args*='')

Plots given *aces-dev* conversion graph using [Graphviz](#) and [pygraphviz](#).

Parameters

- **graph** (DiGraph) – *aces-dev* conversion graph.
- **filename** (unicode) – Filename to use to save the image.
- **prog** (unicode, optional) – {'neato', 'dot', 'twopi', 'circo', 'fdp', 'nop'}, *Graphviz* layout method.
- **args** (unicode, optional) – Additional arguments for *Graphviz*.

Returns *PyGraphviz* graph.

Return type *AGraph*

aces-dev Reference Config Generator

opencolorio_config_aces

<code>generate_config_aces([config_name, ...])</code>	Generates the <i>aces-dev</i> reference implementation <i>OpenColorIO</i> Config using the <i>Mapping</i> method.
---	---

opencolorio_config_aces.generate_config_aces

```
opencolorio_config_aces.generate_config_aces(config_name=None, validate=True, describe=<ColorspaceDescriptionStyle.SHORT_UNION: 14>, config_mapping_file_path=PosixPath('/home/docs/checkouts/readthedocs.org/user_uploads/2019/09/20190920_142138/config-aces/envs/v0.1.0/lib/python3.7/site-packages/opencolorio_config_aces/config/reference/generate/resources/ACES-Config Transforms - Reference Config - Mapping.csv'), analytical=True, additional_data=False)
```

Generates the *aces-dev* reference implementation *OpenColorIO* Config using the *Mapping* method.

The Config generation is constrained by a CSV file exported from the *Reference Config - Mapping* sheet from a [Google Sheets](#) file. The *Google Sheets* file was originally authored using the output of the *aces-dev* conversion graph to support the discussions of the *OpenColorIO Working Group* on the design of the *aces-dev* reference implementation *OpenColorIO* Config. The resulting mapping is the outcome of those discussions and leverages the new *OpenColorIO 2* display architecture while factoring many transforms.

Parameters

- **config_name** (unicode, optional) – *OpenColorIO* config file name, if given the config will be written to disk.
- **validate** (bool, optional) – Whether to validate the config.
- **describe** (int, optional) – Any value from the `opencolorio_config_aces.ColorspaceDescriptionStyle` enum.
- **config_mapping_file_path** (unicode, optional) – Path to the CSV mapping file used by the *Mapping* method.
- **analytical** (bool, optional) – Whether to generate *OpenColorIO* transform families that analytically match the given *ACES CTL* transform, i.e. true to the *aces-dev* reference but not necessarily user friendly.
- **additional_data** (bool, optional) – Whether to return additional data.

Returns *OpenColorIO* config or tuple of *OpenColorIO* config, `opencolorio_config_aces.ConfigData` class instance and dict of *OpenColorIO* colorspaces and `opencolorio_config_aces.config.reference.CTLTransform` class instances.

Return type Config or tuple

Utilities

- *Common*

Common

opencolorio_config_aces.utilities

<code>DocstringDict</code>	A <code>dict</code> sub-class that allows settings a docstring to <code>dict</code> instances.
<code>first_item(iterable[, default])</code>	Returns the first item of given iterable.
<code>common_ancestor(*args)</code>	Returns the common ancestor of given iterables.
<code>paths_common_ancestor(*args)</code>	Returns the common ancestor path from given paths.
<code>vivification()</code>	Implements supports for vivification of the underlying dict like data-structure, magical!
<code>vivified_to_dict(vivified)</code>	Converts given vivified data-structure to dictionary.
<code>message_box(message[, width, padding, ...])</code>	Prints a message inside a box.
<code>is_opencolorio_installed([raise_exception])</code>	Returns if <i>OpenColorIO</i> is installed and available.
<code>REQUIREMENTS_TO_CALLABLE</code>	Mapping of requirements to their respective callables.
<code>required(*requirements)</code>	A decorator checking if various requirements are satisfied.
<code>is_string(a)</code>	Returns if given <i>a</i> variable is a <i>string</i> like variable.
<code>is_iterable(a)</code>	Returns if given <i>a</i> variable is iterable.
<code>git_describe()</code>	Describes the current <i>OpenColorIO Configuration for ACES git</i> version.

opencolorio_config_aces.utilities.DocstringDict

class opencolorio_config_aces.utilities.DocstringDict

A `dict` sub-class that allows settings a docstring to `dict` instances.

`__init__(*args, **kwargs)`

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(*args, **kwargs)</code>	Initialize self.
<code>clear()</code>	
<code>copy()</code>	
<code>fromkeys([value])</code>	Create a new dictionary with keys from iterable and values set to value.
<code>get(key[, default])</code>	Return the value for key if key is in the dictionary, else default.
<code>items()</code>	

continues on next page

Table 8 – continued from previous page

keys()	
pop(k[,d])	If key is not found, d is returned if given, otherwise KeyError is raised
popitem()	2-tuple; but raise KeyError if D is empty.
setdefault(key[, default])	Insert key with a value of default if key is not in the dictionary.
update([E,]**F)	If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
values()	

opencolorio_config_aces.utilities.first_item

opencolorio_config_aces.utilities.**first_item**(iterable, default=None)

Returns the first item of given iterable.

Parameters

- **iterable** (iterable) – Iterable
- **default** (object) – Default value if the iterable is empty.

Returns First iterable item.

Return type object

opencolorio_config_aces.utilities.common_ancestor

opencolorio_config_aces.utilities.**common_ancestor**(*args)

Returns the common ancestor of given iterables.

Other Parameters *args (list, optional) – Iterables to retrieve the common ancestor from.

Returns Common ancestor.

Return type iterable

Examples

```
>>> common_ancestor(('1', '2', '3'), ('1', '2', '0'), ('1', '2', '3', '4'))
('1', '2')
>>> common_ancestor('azerty', 'azetty', 'azello')
'aze'
```


opencolorio_config_aces.utilities.paths_common_ancestor

opencolorio_config_aces.utilities.**paths_common_ancestor**(*args)

Returns the common ancestor path from given paths.

Parameters *args (list, optional) – Paths to retrieve common ancestor from.

Returns Common path ancestor.

Return type unicode

Examples

```
>>> paths_common_ancestor(  
...     '/Users/JohnDoe/Documents', '/Users/JohnDoe/Documents/Test.txt')  
'/Users/JohnDoe/Documents'
```

opencolorio_config_aces.utilities.vivification

opencolorio_config_aces.utilities.**vivification**()

Implements supports for vivification of the underlying dict like data-structure, magical!

Returns

Return type defaultdict

Examples

```
>>> vivified = vivification()  
>>> vivified['my']['attribute'] = 1  
>>> vivified['my']  
defaultdict(<function vivification at 0x...>, {u'attribute': 1})  
>>> vivified['my']['attribute']  
1
```

opencolorio_config_aces.utilities.vivified_to_dict

opencolorio_config_aces.utilities.**vivified_to_dict**(vivified)

Converts given vivified data-structure to dictionary.

Parameters vivified (defaultdict) – Vivified data-structure.

Returns

Return type dict

Examples

```
>>> vivified = vivification()  
>>> vivified['my']['attribute'] = 1  
>>> vivified_to_dict(vivified)  
{u'my': {u'attribute': 1}}
```

opencolorio_config_aces.utilities.message_box

`opencolorio_config_aces.utilities.message_box(message, width=79, padding=3, print_callable=<built-in function print>)`

Prints a message inside a box.

Parameters

- **message** (unicode) – Message to print.
- **width** (`int`, optional) – Message box width.
- **padding** (unicode, optional) – Padding on each sides of the message.
- **print_callable** (callable, optional) – Callable used to print the message box.

Returns Definition success.

Return type `bool`

Examples

```
>>> message = ('Lorem ipsum dolor sit amet, consectetur adipiscing elit, '
...           'sed do eiusmod tempor incididunt ut labore et dolore magna '
...           'aliqua.')
>>> message_box(message, width=75)
=====
*                                     *
*  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do      *
*  eiusmod tempor incididunt ut labore et dolore magna aliqua.          *
*                                     *
=====
True
>>> message_box(message, width=60)
=====
*                                     *
*  Lorem ipsum dolor sit amet, consectetur adipiscing                    *
*  elit, sed do eiusmod tempor incididunt ut labore et                  *
*  dolore magna aliqua.                                                  *
*                                     *
=====
True
>>> message_box(message, width=75, padding=16)
=====
*                                     *
*           Lorem ipsum dolor sit amet, consectetur                    *
*           adipiscing elit, sed do eiusmod tempor                    *
*           incididunt ut labore et dolore magna                    *
*           aliqua.                                                    *
*                                     *
=====
True
```

opencolorio_config_aces.utilities.is_opencolorio_installed

opencolorio_config_aces.utilities.is_opencolorio_installed(*raise_exception=False*)
Returns if *OpenColorIO* is installed and available.

Parameters *raise_exception* (*bool*) – Raise exception if *OpenColorIO* is unavailable.

Returns Is *OpenColorIO* installed.

Return type *bool*

Raises *ImportError* – If *OpenColorIO* is not installed.

opencolorio_config_aces.utilities.REQUIREMENTS_TO_CALLABLE

opencolorio_config_aces.utilities.REQUIREMENTS_TO_CALLABLE = {'NetworkX': <function is_networkx_installed>
Mapping of requirements to their respective callables.

_REQUIREMENTS_TO_CALLABLE [CaseInsensitiveMapping] {'NetworkX', 'OpenImageIO'}

opencolorio_config_aces.utilities.required

opencolorio_config_aces.utilities.required(**requirements*)
A decorator checking if various requirements are satisfied.

Other Parameters **requirements* (*list, optional*) – Requirements to check whether they are satisfied.

Returns

Return type *object*

opencolorio_config_aces.utilities.is_string

opencolorio_config_aces.utilities.is_string(*a*)
Returns if given *a* variable is a *string* like variable.

Parameters *a* (*object*) – Data to test.

Returns Is *a* variable a *string* like variable.

Return type *bool*

Examples

```
>>> is_string("I'm a string!")
True
>>> is_string(["I'm a string!"])
False
```

opencolorio_config_aces.utilities.is_iterable

opencolorio_config_aces.utilities.**is_iterable**(*a*)

Returns if given *a* variable is iterable.

Parameters *a* (**object**) – Variable to check the iterability.

Returns *a* variable iterability.

Return type **bool**

Examples

```
>>> is_iterable([1, 2, 3])
True
>>> is_iterable(1)
False
```

opencolorio_config_aces.utilities.git_describe

opencolorio_config_aces.utilities.**git_describe**()

Describes the current *OpenColorIO Configuration for ACES* git version.

Returns

- `>>> git_describe() # doctest (+SKIP)`
- `'0.1.0'`

Indices and tables

- [genindex](#)
- [search](#)

1.4 ABOUT

OpenColorIO Configuration for ACES by OpenColorIO Contributors

Copyright Contributors to the OpenColorIO Project – ocio-dev@lists.aswf.io

This software is released under terms of New BSD License:

<https://opensource.org/licenses/BSD-3-Clause>

<https://github.com/AcademySoftwareFoundation/OpenColorIO-Config-ACES>

Symbols

`__init__()` (*opencolorio_config_aces.ConfigData* method), 11

`__init__()` (*opencolorio_config_aces.utilities.DocstringDict* method), 19

A

`active_displays` (*opencolorio_config_aces.ConfigData* attribute), 10

`active_views` (*opencolorio_config_aces.ConfigData* attribute), 10

B

`build_aces_conversion_graph()` (in module *opencolorio_config_aces*), 15

C

`classify_aces_ctl_transforms()` (in module *opencolorio_config_aces*), 13

`colorspace_factory()` (in module *opencolorio_config_aces*), 8

`colorspaces` (*opencolorio_config_aces.ConfigData* attribute), 10

`common_ancestor()` (in module *opencolorio_config_aces.utilities*), 20

`ConfigData` (class in *opencolorio_config_aces*), 9

`conversion_path()` (in module *opencolorio_config_aces*), 17

`ctl_transform_to_node()` (in module *opencolorio_config_aces*), 16

D

`default_view_transform` (*opencolorio_config_aces.ConfigData* attribute), 11

`description` (*opencolorio_config_aces.ConfigData* attribute), 10

`discover_aces_ctl_transforms()` (in module *opencolorio_config_aces*), 12

`DocstringDict` (class in *opencolorio_config_aces.utilities*), 19

F

`file_rules` (*opencolorio_config_aces.ConfigData* attribute), 10

`filter_ctl_transforms()` (in module *opencolorio_config_aces*), 14

`filter_nodes()` (in module *opencolorio_config_aces*), 16

`first_item()` (in module *opencolorio_config_aces.utilities*), 20

G

`generate_config()` (in module *opencolorio_config_aces*), 11

`generate_config_aces()` (in module *opencolorio_config_aces*), 18

`git_describe()` (in module *opencolorio_config_aces.utilities*), 24

I

`inactive_colorspaces` (*opencolorio_config_aces.ConfigData* attribute), 11

`is_iterable()` (in module *opencolorio_config_aces.utilities*), 24

`is_opencolorio_installed()` (in module *opencolorio_config_aces.utilities*), 23

`is_string()` (in module *opencolorio_config_aces.utilities*), 23

L

`looks` (*opencolorio_config_aces.ConfigData* attribute), 10

M

`message_box()` (in module *opencolorio_config_aces.utilities*), 22

N

`node_to_ctl_transform()` (in module *opencolorio_config_aces*), 15

P

`paths_common_ancestor()` (in module *opencolorio_config_aces.utilities*), 21

`plot_aces_conversion_graph()` (in module *opencolorio_config_aces*), 17

`print_aces_taxonomy()` (in module *opencolorio_config_aces*), 14

`profile_version` (opencolorio_config_aces.ConfigData attribute), 10

R

`required()` (in module opencolorio_config_aces.utilities), 23

`REQUIREMENTS_TO_CALLABLE` (in module opencolorio_config_aces.utilities), 23

`roles` (opencolorio_config_aces.ConfigData attribute), 10

S

`shared_views` (opencolorio_config_aces.ConfigData attribute), 10

U

`unclassify_ctl_transforms()` (in module opencolorio_config_aces), 13

V

`validate_config()` (in module opencolorio_config_aces), 11

`view_transform_factory()` (in module opencolorio_config_aces), 9

`view_transforms` (opencolorio_config_aces.ConfigData attribute), 10

`viewing_rules` (opencolorio_config_aces.ConfigData attribute), 10

`views` (opencolorio_config_aces.ConfigData attribute), 10

`vivification()` (in module opencolorio_config_aces.utilities), 21

`vivified_to_dict()` (in module opencolorio_config_aces.utilities), 21