
OpenColorIO Configuration for ACES Documentation

Release 1.0.0

OpenColorIO Contributors

Jun 09, 2023

CONTENTS

1	1.1	Features	3
2	1.2	User Guide	5
	2.1	User Guide	5
3	1.3	API Reference	11
	3.1	API Reference	11
4	1.4	About	53
		Index	55

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.

1.1 FEATURES

The following features are available:

- Automatic *OCIO Reference* configuration generation for *aces-dev CTL* reference implementation.
 - Discovery of *aces-dev CTL* transforms.
 - Generation of the *CTL* transforms graph.
 - Spreadsheet-driven generation.
- Generators producing the *OCIO CG* and *Studio* configurations. - Spreadsheet-driven generation.
- Included *CLF* transforms along with generator and discovery support.

1.2 USER GUIDE

2.1 User Guide

The user guide provides an overview of **OpenColorIO Configuration for ACES** and explains important concepts and features, details can be found in the [API Reference](#).

2.1.1 Installation Guide

Cloning the Repository

The *OpenColorIO Configuration for ACES* repository uses [Git submodules](#) thus cloning the repository requires initializing them:

```
git clone --recursive https://github.com/AcademySoftwareFoundation/OpenColorIO-Config-  
↪ACES.git
```

If you have already cloned the repository and forgot the *--recursive* argument, it is possible to initialize the submodules as follows:

```
git submodule update --init --recursive
```

Poetry

The *OpenColorIO Configuration for ACES* repository adopts [Poetry](#) to help managing its dependencies, this is the recommended way to get started with development.

Assuming [python](#) ≥ 3.8 is available on your system along with [OpenColorIO](#) ≥ 2 , the development dependencies are installed with [Poetry](#) as follows:

```
git clone --recursive https://github.com/AcademySoftwareFoundation/OpenColorIO-Config-  
↪ACES.git  
cd OpenColorIO-Config-ACES  
poetry install --extras "optional"
```

The *aces-dev CTL* reference graph can be plotted but it requires [Graphviz](#) to be installed on the system and having installed the optional [pygraphviz](#): python package:

```
poetry install --extras "optional graphviz"
```

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
```

Pypi

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

Primary Dependencies

- `python >= 3.8, < 3.11`
- `black`
- `OpenColorIO`

Optional Dependencies

- `colour`
- `graphviz`
- `jsonpickle`
- `networkx`
- `pygraphviz`

Development Dependencies

- `black`
- `coverage`
- `coveralls`
- `flake8`
- `invoke`
- `mypy`
- `pre-commit`
- `pydata-sphinx-theme`
- `pydocstyle`

- `pytest`
- `pyupgrade`
- `restructuredtext-lint`
- `sphinx >= 4, < 5`
- `twine`

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
```

2.1.2 Usage

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
```

Reference Config

Task	Command
Build	<code>invoke build-config-reference</code>
Build (Docker)	<code>invoke docker-run-build-config-reference</code>
Updating Mapping File	<code>invoke update-mapping-file-reference</code>

CG Config

Task	Command
Build	<code>invoke build-config-cg</code>
Build (Docker)	<code>invoke docker-run-build-config-cg</code>
Updating Mapping File	<code>invoke update-mapping-file-cg</code>

Studio Config

Task	Command
Build	<code>invoke build-config-studio</code>
Build (Docker)	<code>invoke docker-run-build-config-studio</code>
Updating Mapping File	<code>invoke update-mapping-file-studio</code>

2.1.3 Process

aces-dev Discovery & Classification

The config generation process starts with a set of definitions and classes to discover, parse and classify the CTL transforms from `aces-dev`:

- `opencolorio_config_aces.discover_aces_ctl_transforms()`
- `opencolorio_config_aces.classify_aces_ctl_transforms()`
- `opencolorio_config_aces.print_aces_taxonomy()`

Note: This approach allowed us to improve the consistency of the **CTL** transforms `ACEstransformID` while fixing various filename issues.

Using `ACES2065-1` and `OCES` as connection spaces, it is possible to build a colour conversion graph with all the high-level relevant transforms.

- `opencolorio_config_aces.build_aces_conversion_graph()`
- `opencolorio_config_aces.plot_aces_conversion_graph()`

Analytical & Reference Configs Generation

The colour conversion graph is then used to create the *Analytical* config that maps 1-to-1 the **aces-dev** **CTL** transforms to theoretical **OpenColorIO** builtin transforms. This config does not work but it is useful to test basic generation capabilities whilst diagnosing issues in the mapping.

The *Reference* config is driven by a CSV file generated from a [spreadsheet](#) mapping the `ACEstransformID` to **OpenColorIO** builtin transforms.

- `opencolorio_config_aces.generate_config_aces()`

CLF Transforms Generation, Discovery & Classification

The working group decided to express additional colour transforms using **CLF**. Some *CLF* transforms can be serialised into a config which reduces the need for external files. Each *CLF* transform has a `CLFtransformID` specified according to the [CLF Formatting for ACES OCIO Config](#) document.

The repository contains code to generate, discover and classify the additional *CLF* transforms that the *CG* and *Studio* configs require.

- `opencolorio_config_aces.discover_clf_transforms()`
- `opencolorio_config_aces.classify_clf_transforms()`
- `opencolorio_config_aces.print_clf_taxonomy()`

CG Config Generation

The *CG* config generator also uses a CSV file generated from a [spreadsheet](#) that expresses which `ACEstransformID` should be used from the *Reference* config and which `CLFtransformID` should be used from the shipped *CLF* transforms.

- `opencolorio_config_aces.generate_config_cg()`

The conversion process is as follows:

aces-dev Discovery & Classification -> Reference Config Generation -> CLF Transforms Generation, Discovery & Classification -> CG Config Generation

Studio Config Generation

The *Studio* config generator follows the same approach but uses a different [pivot table](#) of the spreadsheet.

- `opencolorio_config_aces.generate_config_studio()`

The conversion process is as follows:

aces-dev Discovery & Classification -> Reference Config Generation -> CLF Transforms Generation, Discovery & Classification -> Studio Config Generation

1.3 API REFERENCE

3.1 API Reference

3.1.1 OpenColorIO Configuration for ACES

Common LUT Format Discovery

opencolorio_config_aces

<code>classify_clf_transforms(...)</code>	Classify given <i>CLF</i> transforms.
<code>discover_clf_transforms([root_directory])</code>	Discover the <i>CLF</i> transform paths in given root directory: The given directory is traversed and the <i>*.clf</i> files are collected.
<code>filter_clf_transforms(clf_transforms[, ...])</code>	Filter given <i>CLF</i> transforms with given filterers.
<code>print_clf_taxonomy()</code>	Print the <i>builtins</i> <i>CLF</i> taxonomy:
<code>unclassify_clf_transforms(...)</code>	Unclassifie given <i>CLF</i> transforms.

opencolorio_config_aces.classify_clf_transforms

opencolorio_config_aces.classify_clf_transforms(*unclassified_clf_transforms*)

Classify given *CLF* transforms.

Parameters

unclassified_clf_transforms (*dict*) – Unclassified *CLF* transforms as returned by `opencolorio_config_aces.discover_clf_transforms()` definition.

Returns

$$\{ "family''_1 : \{ "genus''_1 : \{ \}_{CLF_1}, \dots, "family''_n : \{ "genus''_2 : \{ \}_{CLF_2} \} \}$$

where

$$\{ \}_{CLF_n} = \{ "basename''_n : CLFTransform_n, \dots, "basename''_{n+1} : CLFTransform_{n+1} \}$$

Return type

dict

Examples

```
>>> clf_transforms = classify_clf_transforms(
...     discover_clf_transforms())
>>> family = sorted(clf_transforms.keys())[0]
>>> str(family)
'arri'
>>> genera = sorted(clf_transforms[family])
>>> print(genera)
['Input']
>>> genus = genera[0]
>>> sorted(clf_transforms[family][genus].items())[:2]
[('ARRI.Input.ARRILogC3_Curve_EI800', CLFTransform('arri...input...ARRI.Input.ARRI_
→LogC3_Curve_EI800.clf')), ('ARRI.Input.ARRILogC3_EI800_to_ACES2065-1',
→CLFTransform('arri...input...ARRI.Input.ARRILogC3_EI800_to_ACES2065-1.clf'))]
```

opencolorio_config_aces.discover_clf_transforms

opencolorio_config_aces.**discover_clf_transforms**(root_directory='/home/docs/checkouts/readthedocs.org/user_builds/opencolorio/config-aces/envs/main/lib/python3.9/site-packages/opencolorio_config_aces/clf/transforms')

Discover the *CLF* transform paths in given root directory: The given directory is traversed and the *.*clf* files are collected.

Parameters

root_directory (unicode) – Root directory to traverse to find the *CLF* transforms.

Returns

$$\{ "directory'_1" : ["transform_a.clf", "transform_b.clf"], \dots, "directory'_n" : ["transform_c.clf", "transform_d.clf"] \}$$

Return type

dict

Examples

```
>>> clf_transforms = discover_clf_transforms()
>>> key = sorted(clf_transforms.keys())[0]
>>> os.path.basename(key)
'input'
>>> sorted([os.path.basename(path) for path in clf_transforms[key]]):[:2]
['ARRI.Input.ARRILogC3_Curve_EI800.clf', 'ARRI.Input.ARRILogC3_EI800_to_ACES2065-1.
→clf']
```


opencolorio_config_aces.filter_clf_transforms

`opencolorio_config_aces.filter_clf_transforms(clf_transforms, filterers=None)`

Filter given *CLF* transforms with given filterers.

Parameters

- **clf_transforms** (`dict` or `list`) – *CLF* transforms as returned by `opencolorio_config_aces.classify_clf_transforms()` or `opencolorio_config_aces.unclassify_clf_transforms()` definitions.
- **filterers** (`array_like`, optional) – List of callables used to filter the *CLF* transforms, each callable takes a *CLF* transform as argument and returns whether to include or exclude the *CLF* transform as a bool.

Returns

$[CLFTransform_1, \dots, CLFTransform_n]$

Return type

`list`

Warning:

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

Examples

```
>>> clf_transforms = classify_clf_transforms(
...     discover_clf_transforms())
>>> sorted(
...     filter_clf_transforms(
...         clf_transforms,
...         [lambda x: x.family == 'blackmagic']),
...     key=lambda x: x.path)[0]
CLFTransform('blackmagic...input...BlackmagicDesign.Input.BMDFilm_Gen5_Log-Curve.clf
→')
```

opencolorio_config_aces.print_clf_taxonomy

`opencolorio_config_aces.print_clf_taxonomy()`

Print the *builtins* *CLF* taxonomy:

- The *CLF* transforms are discovered by traversing the directory defined by the `opencolorio_config_aces.clf.reference.ROOT_TRANSFORMS_CLF` attribute using the `opencolorio_config_aces.discover_clf_transforms()` definition.
- The *CLF* transforms are classified by *family* e.g. *aces*, and *genus* e.g. *undefined* using the `opencolorio_config_aces.classify_clf_transforms()` definition.
- The resulting data structure is printed.

opencolorio_config_aces.unclassify_clf_transforms

opencolorio_config_aces.unclassify_clf_transforms(*classified_clf_transforms*)

Unclassifie given *CLF* transforms.

Parameters

classified_clf_transforms (*dict*) – Classified *CLF* transforms as returned by `opencolorio_config_aces.classify_clf_transforms()` definition.

Returns

$[CLFTransform_1, \dots, CLFTransform_n]$

Return type

list

Examples

```
>>> clf_transforms = classify_clf_transforms(  
...     discover_clf_transforms())  
>>> sorted(  
...     unclassify_clf_transforms(clf_transforms), key=lambda x: x.path)[0]  
CLFTransform('arri...input...ARRI.Input.ARRILogC3_Curve_EI800.clf')
```

Common LUT Format Generation

opencolorio_config_aces

<code>generate_clf_transform(filename, transforms, ...)</code>	Take a list of transforms and some metadata and write a <i>CLF</i> transform file.
--	--

opencolorio_config_aces.generate_clf_transform

opencolorio_config_aces.generate_clf_transform(*filename*, *transforms*, *clf_transform_id*, *name*, *input_desc*, *output_desc*, *aces_transform_id=None*, *style=None*)

Take a list of transforms and some metadata and write a *CLF* transform file.

Parameters

- **filename** (*str*) – *CLF* filename.
- **transforms** (*list*) – Transforms to generate the *CLF* transform file for.
- **clf_transform_id** (*str*) – *CLFtransformID*.
- **name** (*str*) – *CLF* transform name.
- **input_desc** (*str*) – *CLF* input descriptor.
- **output_desc** (*str*) – *CLF* output descriptor.
- **aces_transform_id** (*str*, optional) – *ACEStransformID*.
- **style** (*str*, optional) – *OpenColorIO* builtin transform style.

Returns

Updated *GroupTransform*.

Return type

ocio.GroupTransform

Ancillary Objects

opencolorio_config_aces.clf

<code>generate_clf_transforms_arri(output_directory)</code>	Generate CLF files for ARRI color encodings and save to disk.
<code>generate_clf_transforms_bmdfilm(output_directory)</code>	Make the CLF file for BMD-Film_WideGamut_Gen5 plus matrix/curve CLFs.
<code>generate_clf_transforms_canon(output_directory)</code>	Make the CLF file for Canon C-Log3 Cinema Gamut plus matrix/curve CLFs.
<code>generate_clf_transforms_davinci(output_directory)</code>	Make the CLF file for DaVinci Intermediate Wide Gamut plus matrix/curve CLFs.
<code>generate_clf_transforms_itu(output_directory)</code>	Generate OCIO Utility CLF transforms.
<code>generate_clf_transforms_ocio(output_directory)</code>	Generate OCIO Utility CLF transforms.
<code>generate_clf_transforms_panasonic(...)</code>	Make the CLF file for V-Log - V-Gamut plus matrix/curve CLFs.
<code>generate_clf_transforms_red(output_directory)</code>	Make the CLF file for RED Log3G10 REDWideGamutRGB plus matrix/curve CLFs.
<code>generate_clf_transforms_sony(output_directory)</code>	Make all the Sony CLFs.

opencolorio_config_aces.clf.generate_clf_transforms_arriopencolorio_config_aces.clf.**generate_clf_transforms_arri**(*output_directory*)

Generate CLF files for ARRI color encodings and save to disk.

Parameters**output_directory** (PosixPath or WindowsPath) – Directory where generated CLF files should be saved.**Returns**Dictionary of *OpenColorIO Transform* instances.**Return type**

dict

References

-

opencolorio_config_aces.clf.generate_clf_transforms_bmdfilmopencolorio_config_aces.clf.**generate_clf_transforms_bmdfilm**(*output_directory*)

Make the CLF file for BMDFilm_WideGamut_Gen5 plus matrix/curve CLFs.

ReturnsDictionary of *CLF transforms* and *OpenColorIO GroupTransform* instances.**Return type**

dict

References

- Blackmagic Design. (2021). Blackmagic Generation 5 Color Science.

Notes

- The resulting *CLF* transforms were reviewed by *Blackmagic*.

`opencolorio_config_aces.clf.generate_clf_transforms_canon`

`opencolorio_config_aces.clf.generate_clf_transforms_canon(output_directory)`

Make the CLF file for Canon C-Log3 Cinema Gamut plus matrix/curve CLFs.

Returns

Dictionary of *CLF* transforms and *OpenColorIO GroupTransform* instances.

Return type

`dict`

References

- Canon. (2018). White Paper on Canon Log. Retrieved September 22, 2022, from <http://downloads.canon.com/nw/learn/white-papers/cinema-eos/white-paper-canon-log-gamma-curves.pdf>

Notes

- The resulting *CLF* transforms were reviewed by *Canon*.

`opencolorio_config_aces.clf.generate_clf_transforms_davinci`

`opencolorio_config_aces.clf.generate_clf_transforms_davinci(output_directory)`

Make the CLF file for DaVinci Intermediate Wide Gamut plus matrix/curve CLFs.

Returns

Dictionary of *CLF* transforms and *OpenColorIO GroupTransform* instances.

Return type

`dict`

References

- Blackmagic Design. (2020). Wide Gamut Intermediate DaVinci Resolve. Retrieved December 12, 2020, from https://documents.blackmagicdesign.com/InformationNotes/DaVinci_Resolve_17_Wide_Gamut_Intermediate.pdf?v=1607414410000

Notes

- The resulting *CLF* transforms were reviewed by *Blackmagic*.

`opencolorio_config_aces.clf.generate_clf_transforms_itu`

`opencolorio_config_aces.clf.generate_clf_transforms_itu(output_directory)`

Generate OCIO Utility CLF transforms.

`opencolorio_config_aces.clf.generate_clf_transforms_ocio`

`opencolorio_config_aces.clf.generate_clf_transforms_ocio(output_directory)`

Generate OCIO Utility CLF transforms.

`opencolorio_config_aces.clf.generate_clf_transforms_panasonic`

`opencolorio_config_aces.clf.generate_clf_transforms_panasonic(output_directory)`

Make the CLF file for V-Log - V-Gamut plus matrix/curve CLFs.

Returns

Dictionary of *CLF* transforms and *OpenColorIO GroupTransform* instances.

Return type

`dict`

References

- Panasonic. (2014). VARICAM V-Log/V-Gamut (pp. 1–7). http://pro-av.panasonic.net/en/varicam/common/pdf/VARICAM_V-Log_V-Gamut.pdf

Notes

- The resulting *CLF* transforms were reviewed by *Panasonic*.

`opencolorio_config_aces.clf.generate_clf_transforms_red`

`opencolorio_config_aces.clf.generate_clf_transforms_red(output_directory)`

Make the CLF file for RED Log3G10 REDWideGamutRGB plus matrix/curve CLFs.

Returns

Dictionary of *CLF* transforms and *OpenColorIO GroupTransform* instances.

Return type

`dict`

References

- RED Digital Cinema. (2017). White Paper on REDWideGamutRGB and Log3G10. Retrieved January 16, 2021, from <https://www.red.com/download/white-paper-on-redwidegamutrgb-and-log3g10>

Notes

- The resulting *CLF* transforms were reviewed by *RED*.

opencolorio_config_aces.clf.generate_clf_transforms_sony

opencolorio_config_aces.clf.generate_clf_transforms_sony(output_directory)

Make all the Sony CLFs.

Returns

Dictionary of *CLF* transforms and *OpenColorIO GroupTransform* instances.

Return type

dict

Notes

- The full transforms generated here were developed in collaboration with Sony.

Generation

Profile Version

opencolorio_config_aces

ProfileVersion([major, minor])	Define the data container for an <i>OpenColorIO</i> config profile version.
PROFILE_VERSION_DEFAULT	Define the data container for an <i>OpenColorIO</i> config profile version.
SUPPORTED_PROFILE_VERSIONS	Built-in mutable sequence.

opencolorio_config_aces.ProfileVersion

class opencolorio_config_aces.ProfileVersion(major=None, minor=None)

Define the data container for an *OpenColorIO* config profile version.

Parameters

- **major** (int, optional) – Major version number.
- **minor** (int, optional) – Minor version number.

major

minor

Examples

```
>>> profile_version = ProfileVersion()
>>> profile_version.parse("2.0")
>>> profile_version.major
2
>>> profile_version.minor
0
>>> profile_version_1 = PROFILE_VERSION_DEFAULT
>>> profile_version_2 = ProfileVersion(2, 1)
>>> profile_version_2 > profile_version_1
True
```

```
__init__(major=None, minor=None)
```

Methods

```
__init__([major, minor])
```

```
parse(vstring)
```

Attributes

major	Getter property for the major version number.
minor	Getter property for the minor version number.
version_re	

opencolorio_config_aces.PROFILE_VERSION_DEFAULT

```
opencolorio_config_aces.PROFILE_VERSION_DEFAULT = ProfileVersion ('2.0')
```

Define the data container for an *OpenColorIO* config profile version.

Parameters

- **major** (`int`, optional) – Major version number.
- **minor** (`int`, optional) – Minor version number.

```
opencolorio_config_aces.major
```

```
opencolorio_config_aces.minor
```

Examples

```
>>> profile_version = ProfileVersion()
>>> profile_version.parse("2.0")
>>> profile_version.major
2
>>> profile_version.minor
0
>>> profile_version_1 = PROFILE_VERSION_DEFAULT
>>> profile_version_2 = ProfileVersion(2, 1)
>>> profile_version_2 > profile_version_1
True
```

opencolorio_config_aces.SUPPORTED_PROFILE_VERSIONS

```
opencolorio_config_aces.SUPPORTED_PROFILE_VERSIONS = [ProfileVersion ('2.0'),
ProfileVersion ('2.1')]
```

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

Config Generation Common Objects

opencolorio_config_aces

<code>ConfigData(schema_version, profile_version, ...)</code>	Define the data container for an <i>OpenColorIO</i> config.
<code>deserialize_config_data(path)</code>	Deserialize the <i>JSON OpenColorIO</i> config data container at given path.
<code>generate_config(data[, config_name, ...])</code>	Generate the <i>OpenColorIO</i> config from given data.
<code>serialize_config_data(data, path)</code>	Serialize the <i>OpenColorIO</i> config data container as a <i>JSON</i> file.
<code>validate_config(config)</code>	Validate given <i>OpenColorIO</i> config.

opencolorio_config_aces.ConfigData

```
class opencolorio_config_aces.ConfigData(schema_version: ~opencolorio_config_aces.config.generation.profile_version.ProfileVersion
= <factory>, profile_version: ~opencolorio_config_aces.config.generation.profile_version.ProfileVersion
= <factory>, name: str = <factory>, description: str =
'An "OpenColorIO" config generated by
"OpenColorIO-Config-ACES".', search_path: list =
<factory>, roles: dict = <factory>, colorspaces: list =
<factory>, named_transforms: list = <factory>,
view_transforms: list = <factory>, looks: list =
<factory>, shared_views: list = <factory>, views: list =
<factory>, active_displays: list = <factory>,
active_views: list = <factory>, file_rules: list =
<factory>, viewing_rules: list = <factory>,
inactive_colorspaces: list = <factory>,
default_view_transform: str = <factory>)
```


Define the data container for an *OpenColorIO* config.

Parameters

- **profile_version** ([ProfileVersion](#), optional) – Config major and minor version, i.e. (1, 0) or (2, 0).
- **name** (unicode, optional) – Config name.
- **description** (unicode, optional) – Config description.
- **search_path** ([list](#), optional) – Config search path.
- **roles** ([dict](#)) – Config roles, a dict of role and *Colorspace* name.
- **colorspaces** (array_like) – Config colorspaces, an iterable of `PyOpenColorIO.ColorSpace` class instances or mappings to create them with `opencolorio_config_aces.colorspace_factory()` definition.
- **named_transforms** (array_like) – Config *NamedTransform*'s, an iterable of :attr: `PyOpenColorIO.NamedTransform` class instances or mappings to create them with `opencolorio_config_aces.named_transform_factory()` definition.
- **view_transforms** (array_like, optional) – Config view transforms, an iterable of `PyOpenColorIO.ViewTransform` class instances or mappings to create them with `opencolorio_config_aces.view_transform_factory()` definition.
- **looks** (array_like, optional) – Config looks, an iterable of `PyOpenColorIO.Look` class instances or mappings to create them with `opencolorio_config_aces.look_factory()` definition.
- **shared_views** (array_like, optional) – Config shared views, an iterable of dicts of view, *ViewTransform*, *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.

schema_version

Type

`opencolorio_config_aces.config.generation.profile_version.ProfileVersion`

profile_version

Type

`opencolorio_config_aces.config.generation.profile_version.ProfileVersion`

name
Type
str

description
Type
str

search_path
Type
list

roles
Type
dict

colorspaces
Type
list

named_transforms
Type
list

view_transforms
Type
list

looks
Type
list

shared_views
Type
list

views
Type
list

active_displays
Type
list

active_views
Type
list

file_rules
Type
list

viewing_rules

Type
list

inactive_colorspaces

Type
list

default_view_transform

Type
str

```
__init__(schema_version: ~opencolorio_config_aces.config.generation.profile_version.ProfileVersion
         = <factory>, profile_version:
         ~opencolorio_config_aces.config.generation.profile_version.ProfileVersion = <factory>,
         name: str = <factory>, description: str = 'An "OpenColorIO" config generated by
         "OpenColorIO-Config-ACES".', search_path: list = <factory>, roles: dict = <factory>,
         colorspaces: list = <factory>, named_transforms: list = <factory>, view_transforms: list
         = <factory>, looks: list = <factory>, shared_views: list = <factory>, views: list =
         <factory>, active_displays: list = <factory>, active_views: list = <factory>, file_rules:
         list = <factory>, viewing_rules: list = <factory>, inactive_colorspaces: list =
         <factory>, default_view_transform: str = <factory>) → None
```

Methods

```
__init__([schema_version,    profile_version,
...])
```

Attributes

description
schema_version
profile_version
name
search_path
roles
colorspaces
named_transforms
view_transforms
looks
shared_views
views
active_displays
active_views
file_rules
viewing_rules
inactive_colorspaces
default_view_transform

opencolorio_config_aces.deserialize_config_data

opencolorio_config_aces.**deserialize_config_data**(*path*)

Deserialize the *JSON OpenColorIO* config data container at given path.

Parameters

path (unicode) – *JSON* file path.

Returns

Deserialized *JSON OpenColorIO* config data container.

Return type

ConfigData

opencolorio_config_aces.generate_config

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

Generate 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.
- **base_config** (bool, optional) – *OpenColorIO* base config inherited for initial data.

Returns

OpenColorIO config.

Return type

Config

opencolorio_config_aces.serialize_config_data

opencolorio_config_aces.**serialize_config_data**(data, path)

Serialize the *OpenColorIO* config data container as a *JSON* file.

Parameters

- **data** (*ConfigData*) – *OpenColorIO* config data container to serialize.
- **path** (unicode) – *JSON* file path.

opencolorio_config_aces.validate_config

opencolorio_config_aces.**validate_config**(config)

Validate given *OpenColorIO* config.

Parameters

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

Returns

Whether the *OpenColorIO* config is valid.

Return type

bool

Factories

opencolorio_config_aces

BUILTIN_TRANSFORMS	Mapping of <i>OpenColorIO</i> builtintransforms to their profile version.
TRANSFORM_FACTORIES	<i>OpenColorIO</i> transform factories.
colorspace_factory(name[, family, encoding, ...])	<i>OpenColorIO Colorspace</i> factory.
group_transform_factory(transforms)	<i>OpenColorIO GroupTransform</i> factory.
look_factory(name[, process_space, ...])	<i>OpenColorIO Look</i> factory.
named_transform_factory(name[, family, ...])	<i>OpenColorIO NamedTransform</i> factory.
produce_transform(transform)	Produce given transform.
transform_factory(**kwargs)	<i>OpenColorIO</i> transform factory.
view_transform_factory(name[, family, ...])	<i>OpenColorIO ViewTransform</i> factory.

`opencolorio_config_aces.BUILTIN_TRANSFORMS`

```

opencolorio_config_aces.BUILTIN_TRANSFORMS = {'ACES-LMT - ACES 1.3 Reference Gamut
Compression': ProfileVersion ('2.1'), 'ACES-LMT - BLUE_LIGHT_ARTIFACT_FIX': ProfileVersion
('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 - HDR-CINEMA-108nit-7.2nit-P3lim_1.1':
ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 -
HDR-VIDEO-1000nit-15nit-P3lim_1.1': ProfileVersion ('2.0'), 'ACES-OUTPUT -
ACES2065-1_to_CIE-XYZ-D65 - HDR-VIDEO-1000nit-15nit-REC2020lim_1.1': ProfileVersion
('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 - HDR-VIDEO-2000nit-15nit-P3lim_1.1':
ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 -
HDR-VIDEO-2000nit-15nit-REC2020lim_1.1': ProfileVersion ('2.0'), 'ACES-OUTPUT -
ACES2065-1_to_CIE-XYZ-D65 - HDR-VIDEO-4000nit-15nit-P3lim_1.1': ProfileVersion ('2.0'),
'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 - HDR-VIDEO-4000nit-15nit-REC2020lim_1.1':
ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 -
SDR-CINEMA-D60sim-D65_1.1': ProfileVersion ('2.0'), 'ACES-OUTPUT -
ACES2065-1_to_CIE-XYZ-D65 - SDR-CINEMA-D60sim-DCI_1.0': ProfileVersion ('2.0'),
'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 - SDR-CINEMA-D65sim-DCI_1.1': ProfileVersion
('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 - SDR-CINEMA-REC709lim_1.1':
ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 - SDR-CINEMA_1.0':
ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 -
SDR-VIDEO-D60sim-D65_1.0': ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65
- SDR-VIDEO-P3lim_1.1': ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65 -
SDR-VIDEO-REC709lim_1.1': ProfileVersion ('2.0'), 'ACES-OUTPUT - ACES2065-1_to_CIE-XYZ-D65
- SDR-VIDEO_1.0': ProfileVersion ('2.0'), 'ACEScc_to_ACES2065-1': ProfileVersion ('2.0'),
'ACEScct_to_ACES2065-1': ProfileVersion ('2.0'), 'ACEScg_to_ACES2065-1': ProfileVersion
('2.0'), 'ACESproxy10i_to_ACES2065-1': ProfileVersion ('2.0'), 'ADX10_to_ACES2065-1':
ProfileVersion ('2.0'), 'ADX16_to_ACES2065-1': ProfileVersion ('2.0'),
'ARRI_ALEXA-LOGC-EI800-AWG_to_ACES2065-1': ProfileVersion ('2.0'),
'ARRI_LOGC4_to_ACES2065-1': ProfileVersion ('2.0'), 'CANON_CLOG2-CGAMUT_to_ACES2065-1':
ProfileVersion ('2.0'), 'CANON_CLOG3-CGAMUT_to_ACES2065-1': ProfileVersion ('2.0'), 'CURVE
- ACEScct-LOG_to_LINEAR': ProfileVersion ('2.0'), 'CURVE - CANON_CLOG2_to_LINEAR':
ProfileVersion ('2.0'), 'CURVE - CANON_CLOG3_to_LINEAR': ProfileVersion ('2.0'), 'CURVE -
LINEAR_to_ST-2084': ProfileVersion ('2.0'), 'CURVE - ST-2084_to_LINEAR': ProfileVersion
('2.0'), 'DISPLAY - CIE-XYZ-D65_to_G2.2-REC.709': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_G2.6-P3-D60-BFD': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_G2.6-P3-D65': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_G2.6-P3-DCI-BFD': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_REC.1886-REC.2020': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_REC.1886-REC.709': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_REC.2100-HLG-1000nit': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_REC.2100-PQ': ProfileVersion ('2.0'), 'DISPLAY -
CIE-XYZ-D65_to_ST2084-P3-D65': ProfileVersion ('2.0'), 'DISPLAY - CIE-XYZ-D65_to_sRGB':
ProfileVersion ('2.0'), 'IDENTITY': ProfileVersion ('2.0'),
'PANASONIC_VLOG-VGAMUT_to_ACES2065-1': ProfileVersion ('2.0'),
'RED_LOG3G10-RWG_to_ACES2065-1': ProfileVersion ('2.0'),
'RED_REDLOGFILM-RWG_to_ACES2065-1': ProfileVersion ('2.0'),
'SONY_SLOG3-SGAMUT3-VENICE_to_ACES2065-1': ProfileVersion ('2.0'),
'SONY_SLOG3-SGAMUT3.CINE-VENICE_to_ACES2065-1': ProfileVersion ('2.0'),
'SONY_SLOG3-SGAMUT3.CINE_to_ACES2065-1': ProfileVersion ('2.0'),
'SONY_SLOG3-SGAMUT3_to_ACES2065-1': ProfileVersion ('2.0'), 'UTILITY -
ACES-AP0_to_CIE-XYZ-D65_BFD': ProfileVersion ('2.0'), 'UTILITY -
ACES-AP1_to_CIE-XYZ-D65_BFD': ProfileVersion ('2.0'), 'UTILITY -
ACES-AP1_to_LINEAR-REC709_BFD': ProfileVersion ('2.0')}}

```

Mapping of *OpenColorIO* builtintransforms to their profile version.

BUILTIN_TRANSFORMS : dict

opencolorio_config_aces.TRANSFORM_FACTORIES

```
opencolorio_config_aces.TRANSFORM_FACTORIES = {'CLF Transform to Group Transform':  
<function transform_factory_clf_transform_to_group_transform>, 'Constructor': <function  
transform_factory_constructor>, 'Setter': <function transform_factory_setter>}
```

OpenColorIO transform factories.

TRANSFORM_FACTORIES : dict

opencolorio_config_aces.colorspace_factory

```
opencolorio_config_aces.colorspace_factory(name, family=None, encoding=None, aliases=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, **kwargs)
```

OpenColorIO Colorspace factory.

Parameters

- **name** (unicode) – *OpenColorIO* Colorspace name.
- **family** (unicode, optional) – *OpenColorIO* Colorspace family.
- **encoding** (unicode, optional) – *OpenColorIO* Colorspace encoding.
- **aliases** (unicode or array_like, optional) – *OpenColorIO* Colorspace aliases.
- **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** (dict or object, optional) – *To Reference OpenColorIO* transform.
- **from_reference** (dict or object, optional) – *From Reference OpenColorIO* transform.
- **reference_space** (unicode or ReferenceSpaceType, optional) – *OpenColorIO* Colorspace reference space.
- **is_data** (bool, optional) – Whether the *Colorspace* represents data.
- **base_colorspace** (dict or ColorSpace, optional) – *OpenColorIO* base *Colorspace* inherited for initial attribute values.
- ****kwargs** (dict, optional) – Keywords arguments.

Returns

OpenColorIO colorspace.

Return type

ocio.ColorSpace

opencolorio_config_aces.group_transform_factory

opencolorio_config_aces.group_transform_factory(transforms)

OpenColorIO GroupTransform factory.

Parameters

transforms (array_like) – *OpenColorIO transforms.*

Returns

OpenColorIO GroupTransform.

Return type

ocio.GroupTransform

opencolorio_config_aces.look_factory

opencolorio_config_aces.look_factory(name, process_space=None, description=None,
forward_transform=None, inverse_transform=None,
base_look=None, **kwargs)

OpenColorIO Look factory.

Parameters

- **name** (unicode) – *OpenColorIO Look name.*
- **process_space** (unicode, optional) – *OpenColorIO Look process space, e.g. OpenColorIO Colorspace or role name.*
- **description** (unicode, optional) – *OpenColorIO Look description.*
- **forward_transform** (dict or object, optional) – *To Reference OpenColorIO Look transform.*
- **inverse_transform** (dict or object, optional) – *From Reference OpenColorIO Look transform.*
- **base_look** (dict or ViewTransform, optional) – *OpenColorIO base Look inherited for initial attribute values.*
- ****kwargs** (dict, optional) – *Keywords arguments.*

Returns

OpenColorIO Look.

Return type

ocio.Look

opencolorio_config_aces.named_transform_factory

opencolorio_config_aces.named_transform_factory(name, family=None, encoding=None,
aliases=None, categories=None,
description=None, forward_transform=None,
inverse_transform=None,
base_named_transform=None, **kwargs)

OpenColorIO NamedTransform factory.

Parameters

- **name** (unicode) – *OpenColorIO NamedTransform name.*
- **family** (unicode, optional) – *OpenColorIO NamedTransform family.*
- **encoding** (unicode, optional) – *OpenColorIO NamedTransform encoding.*

- **aliases** (unicode or array_like, optional) – *OpenColorIO NamedTransform* aliases.
- **categories** (unicode or array_like, optional) – *OpenColorIO NamedTransform* categories.
- **description** (unicode, optional) – *OpenColorIO NamedTransform* description.
- **forward_transform** (dict or object, optional) – *Forward OpenColorIO* transform.
- **inverse_transform** (dict or object, optional) – *Inverse OpenColorIO* transform.
- **base_named_transform** (dict or NamedTransform, optional) – *OpenColorIO* base *NamedTransform* inherited for initial attribute values.
- ****kwargs** (dict, optional) – Keywords arguments.

Returns

OpenColorIO NamedTransform.

Return type

ocio.NamedTransform

opencolorio_config_aces.produce_transform

opencolorio_config_aces.**produce_transform**(transform)

Produce given transform.

Parameters

transform (object or dict or array_like) – Transform to produce, either a single transform if a *Mapping* instance or a *GroupTransform* is a *Sequence* instance.

Returns

OpenColorIO transform.

Return type

object

opencolorio_config_aces.transform_factory

opencolorio_config_aces.**transform_factory**(**kwargs)

OpenColorIO transform factory.

Parameters

- **factory** (unicode) – {'Default', 'CLF Transform to *GroupTransform*'}, *OpenColorIO* transform factory name.
- ****kwargs** (dict, optional) – Keywords arguments for the factory.

Returns

OpenColorIO transform.

Return type

object

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, **kwargs)`

OpenColorIO ViewTransform factory.

Parameters

- **name** (unicode) – *OpenColorIO ViewTransform name.*
- **family** (unicode, optional) – *OpenColorIO ViewTransform family.*
- **categories** (array_like, optional) – *OpenColorIO ViewTransform categories.*
- **description** (unicode, optional) – *OpenColorIO ViewTransform description.*
- **to_reference** (dict or object, optional) – *To Reference OpenColorIO ViewTransform.*
- **from_reference** (dict or object, optional) – *From Reference OpenColorIO ViewTransform.*
- **reference_space** (unicode or ReferenceSpaceType, optional) – *OpenColorIO ViewTransform reference space.*
- **base_view_transform** (dict or ViewTransform, optional) – *OpenColorIO base ViewTransform inherited for initial attribute values.*
- ****kwargs** (dict, optional) – *Keywords arguments.*

Returns

OpenColorIO ViewTransform.

Return type

`ocio.ViewTransform`

Reference Configuration

aces-dev Discovery

`opencolorio_config_aces`

<code>version_aces_dev()</code>	Return the current <i>aces-dev</i> version, trying first with <i>git</i> , then by parsing the <i>CHANGELOG.md</i> file.
<code>classify_aces_ctl_transforms(...)</code>	Classifie given <i>ACES CTL</i> transforms.
<code>discover_aces_ctl_transforms([root_directory])</code>	Discover the <i>ACES CTL</i> transform paths in given root directory: The given directory is traversed and the <i>*.ctl</i> files are collected.
<code>filter_ctl_transforms(ctl_transforms[, ...])</code>	Filter given <i>ACES CTL</i> transforms with given filterers.
<code>print_aces_taxonomy()</code>	Print <i>aces-dev</i> taxonomy:
<code>unclassify_ctl_transforms(...)</code>	Unclassifie given <i>ACES CTL</i> transforms.

opencolorio_config_aces.version_aces_dev

opencolorio_config_aces.version_aces_dev()

Return the current *aces-dev* version, trying first with *git*, then by parsing the *CHANGELOG.md* file.

Returns

aces-dev version.

Return type

`str`

opencolorio_config_aces.classify_aces_ctl_transforms

opencolorio_config_aces.classify_aces_ctl_transforms(*unclassified_ctl_transforms*)

Classifie 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.ACEScc_
↪ to_ACES.ctl')))]
```

opencolorio_config_aces.discover_aces_ctl_transforms

`opencolorio_config_aces.discover_aces_ctl_transforms(root_directory='/home/docs/checkouts/readthedocs.org/user_configs/aces/envs/main/lib/python3.9/site-packages/opencolorio_config_aces/config/reference/aces-dev/transforms/ctl')`

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

Parameters

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

Returns

$$\{ "directory_1" : ["transform_a.ctl", "transform_b.ctl"], \dots, "directory_n" : ["transform_c.ctl", "transform_d.ctl"] \}$$

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.filter_ctl_transforms

`opencolorio_config_aces.filter_ctl_transforms(ctl_transforms, filterers=None)`

Filter 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**()

Print *aces-dev* taxonomy:

- The *aces-dev* CTL transforms are discovered by traversing the directory defined by the `opencolorio_config_aces.config.reference.ROOT_TRANSFORMS_CTL` 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 data structure is printed.

opencolorio_config_aces.unclassify_ctl_transforms

opencolorio_config_aces.**unclassify_ctl_transforms**(*classified_ctl_transforms*)

Unclassifie 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')
```

aces-dev Conversion Graph

opencolorio_config_aces

<code>build_aces_conversion_graph(ctl_transforms)</code>	Build the <i>aces-dev</i> conversion graph from given <i>ACES CTL</i> transforms.
<code>conversion_path(graph, source, target)</code>	Return the conversion path from the source node to the target node in the <i>aces-dev</i> conversion graph.
<code>ctl_transform_to_node(graph, ctl_transform)</code>	Return the node name from given <i>ACES CTL</i> transform.
<code>filter_nodes(graph[, filterers])</code>	Filter given <i>aces-dev</i> conversion graph nodes with given filterers.
<code>node_to_ctl_transform(graph, node)</code>	Return the <i>ACES CTL</i> transform from given node name.
<code>plot_aces_conversion_graph(graph, filename)</code>	Plot 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*)

Build 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.conversion_path

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

Return 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.ctl_transform_to_node

`opencolorio_config_aces.ctl_transform_to_node(graph, ctl_transform)`

Return 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)`

Filter 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.node_to_ctl_transform

opencolorio_config_aces.**node_to_ctl_transform**(graph, node)

Return 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.plot_aces_conversion_graph

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

Plot 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>ColorspaceDescriptionStyle(value)</code>	Enum storing the various <i>OpenColorIO Colorspace</i> description styles.
<code>version_config_mapping_file([path])</code>	Return the current version of given <i>CSV</i> mapping file.
<code>generate_config_aces([config_name, ...])</code>	Generate the <i>aces-dev</i> reference implementation <i>OpenColorIO</i> config using the <i>Mapping</i> method.

opencolorio_config_aces.ColorspaceDescriptionStyle

class opencolorio_config_aces.ColorspaceDescriptionStyle(value)
Enum storing the various *OpenColorIO Colorspace* description styles.
`__init__()`

Attributes

NONE
ACES
OPENCOLORIO
SHORT
LONG
SHORT_UNION
LONG_UNION

opencolorio_config_aces.version_config_mapping_file

opencolorio_config_aces.version_config_mapping_file(path=PosixPath('/home/docs/checkouts/readthedocs.org/user_uploads/2020/07/20200720_01/config-aces/envs/main/lib/python3.9/site-packages/opencolorio_config_aces/config/reference/generate/reference-config-aces-reference-transforms-v1.0.0-reference-config-mapping.csv'))

Return the current version of given *CSV* mapping file.

No parsing of the file content is perform, a simple regex is used to extract the version of the file name.

Parameters

path (Path or `str`, optional) – Path to the *CSV* mapping file.

Returns

CSV mapping file version.

Return type

`str`

Examples

```
>>> path = (
...     "/tmp/OpenColorIO-Config-ACES Reference Transforms - v0.1.0 - "
...     "Reference Config - Mapping.csv"
... )
>>> version_config_mapping_file(path)
'v0.1.0'
>>> path = (
...     "/tmp/OpenColorIO-Config-ACES Reference Transforms - "
...     "Reference Config - Mapping.csv"
... )
>>> version_config_mapping_file(path)
''
```

opencolorio_config_aces.generate_config_aces

```
opencolorio_config_aces.generate_config_aces(config_name=None,
                                             profile_version=ProfileVersion('2.0'), validate=True,
                                             describe=ColorspaceDescriptionStyle.SHORT_UNION,
                                             config_mapping_file_path=PosixPath('/home/docs/checkouts/readthedocs.org/user_uploads/2020/09/20200920_154534/opencolorio_config_aces/config/reference/generate/resources/Config-ACES Reference Transforms - v1.0.0 - Reference Config - Mapping.csv'), analytical=True,
                                             scheme='Modern 1', additional_data=False)
```

Generate 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.
- **profile_version** ([ProfileVersion](#), optional) – *OpenColorIO* config profile version.
- **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.
- **scheme** (str, optional) – {“Legacy”, “Modern 1”}, Naming convention scheme to use.
- **additional_data** (bool, optional) – Whether to return additional data.

Returns

OpenColorIO config or tuple of *OpenColorIO* config and `opencolorio_config_aces.ConfigData` class instance.

Return type

Config or `tuple`

ACES Computer Graphics (CG) Config Generator

`opencolorio_config_aces`

<code>generate_config_cg([data, config_name, ...])</code>	Generate the <i>ACES</i> Computer Graphics (CG) <i>OpenColorIO</i> config.
---	--

`opencolorio_config_aces.generate_config_cg`

```
opencolorio_config_aces.generate_config_cg(data=None, config_name=None,
                                           profile_version=ProfileVersion('2.0'), validate=True,
                                           describe=ColorspaceDescriptionStyle.SHORT_UNION,
                                           config_mapping_file_path=PosixPath('/home/docs/checkouts/readthedocs
                                           config-aces/envs/main/lib/python3.9/site-
                                           packages/opencolorio_config_aces/config/cg/generate/resources/OpenCol
                                           Config-ACES CG and Studio Transforms - v1.0.0 - CG
                                           Config - Mapping.csv'), scheme='Modern 1',
                                           additional_data=False)
```

Generate the *ACES* Computer Graphics (CG) *OpenColorIO* config.

The default process is as follows:

- The *ACES* CG *OpenColorIO* config generator invokes the *aces-dev* reference implementation *OpenColorIO* config generator via the `opencolorio_config_aces.generate_config_aces()` definition and the default reference config mapping file.
- The *ACES* CG *OpenColorIO* config generator filters and extends the data from the *aces-dev* reference implementation *OpenColorIO* config with the given CG config mapping file:
 - The builtin *CLF* transforms are discovered and classified.
 - The CG config mapping file is parsed.
 - The list of implicit colorspaces is built, e.g. *ACES2065-1*, *Raw*, etc. . .
 - The colorspaces, looks and view transforms are filtered according to the parsed CG config mapping file data.
 - The displays, views, and shared views are filtered similarly.
 - The active displays and views are also filtered.
 - The builtin *CLF* transforms are filtered according to the parsed CG config mapping file data and converted to colorspaces (or named transforms).
 - Finally, the roles and aliases are updated.

Parameters

- **data** (`ConfigData`, optional) – *OpenColorIO* config data to derive the config from, the default is to use the *aces-dev* reference implementation *OpenColorIO* config.
- **config_name** (unicode, optional) – *OpenColorIO* config file name, if given the config will be written to disk.

- **profile_version** (`ProfileVersion`, optional) – *OpenColorIO* config profile version.
- **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 to describe the transforms mapping.
- **scheme** (`str`, optional) – {“Legacy”, “Modern 1”}, Naming convention scheme to use.
- **additional_data** (`bool`, optional) – Whether to return additional data.

Returns

OpenColorIO config or tuple of *OpenColorIO* config and `opencolorio_config_aces.ConfigData` class instance.

Return type

Config or `tuple`

ACES Studio Config Generator

`opencolorio_config_aces`

```
generate_config_studio([data, config_name, Generate the ACES Studio OpenColorIO config.
...])
```

opencolorio_config_aces.generate_config_studio

```
opencolorio_config_aces.generate_config_studio(data=None, config_name=None,
                                                profile_version=ProfileVersion('2.0'),
                                                validate=True, describe=ColorspaceDescriptionStyle.SHORT_UNION,
                                                config_mapping_file_path=PosixPath('/home/docs/checkouts/readth
config-aces/envs/main/lib/python3.9/site-
packages/opencolorio_config_aces/config/studio/generate/resources/
Config-ACES CG and Studio Transforms - v1.0.0 -
Studio Config - Mapping.csv'), scheme='Modern 1',
                                                additional_data=False)
```

Generate the *ACES Studio OpenColorIO* config.

The default process is as follows:

- The *ACES Studio OpenColorIO* config generator invokes the *ACES CG OpenColorIO* config generator with the given studio config mapping file via the `opencolorio_config_aces.generate_config_cg()` definition.
- The *ACES CG OpenColorIO* config generator invokes the *aces-dev* reference implementation *OpenColorIO* config generator via the `opencolorio_config_aces.generate_config_aces()` definition and the default reference config mapping file.
- With the data from the *aces-dev* reference implementation *OpenColorIO* config generated, the *ACES CG OpenColorIO* config generator produces the *ACES Studio OpenColorIO* config by filtering and extending it with the given studio config mapping file.

Parameters

- **data** (`ConfigData`, optional) – *OpenColorIO* config data to derive the config from, the default is to use the *ACES CG OpenColorIO* config.
- **config_name** (unicode, optional) – *OpenColorIO* config file name, if given the config will be written to disk.
- **profile_version** (`ProfileVersion`, optional) – *OpenColorIO* config profile version.
- **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 to describe the transforms mapping.
- **scheme** (`str`, optional) – {“Legacy”, “Modern 1”}, Naming convention scheme to use.
- **additional_data** (`bool`, optional) – Whether to return additional data.

Returns

OpenColorIO config or tuple of *OpenColorIO* config and `opencolorio_config_aces.ConfigData` class instance.

Return type

Config or `tuple`

Utilities**Common**

`opencolorio_config_aces.utilities`

<code>ROOT_BUILD_DEFAULT</code>	Path subclass for non-Windows systems.
<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>	Return the first item of given iterable.
<code>common_ancestor(*args)</code>	Return the common ancestor of given iterables.
<code>paths_common_ancestor(*args)</code>	Return the common ancestor path from given paths.
<code>vivification()</code>	Implement supports for vivification of the underlying dict like data-structure, magical!
<code>vivified_to_dict(vivified)</code>	Convert given vivified data-structure to dictionary.
<code>message_box(message[, width, padding, ...])</code>	Print a message inside a box.
<code>is_colour_installed([raise_exception])</code>	Return if <i>Colour</i> is installed and available.
<code>is_jsonpickle_installed([raise_exception])</code>	Return if <i>jsonpickle</i> is installed and available.
<code>is_networkx_installed([raise_exception])</code>	Return if <i>NetworkX</i> is installed and available.
<code>REQUIREMENTS_TO_CALLABLE</code>	Mapping of requirements to their respective callables.
<code>required(*requirements)</code>	Decorate a function to check whether various ancillary package requirements are satisfied.
<code>is_string(a)</code>	Return if given <i>a</i> variable is a <i>string</i> like variable.
<code>is_iterable(a)</code>	Return if given <i>a</i> variable is iterable.
<code>git_describe()</code>	Describe the current <i>OpenColorIO Configuration for ACES git</i> version.
<code>matrix_3x3_to_4x4(M)</code>	Convert given 3x3 matrix <i>M</i> to a raveled 4x4 matrix.
<code>multi_replace(name, patterns)</code>	Update given name by applying in succession the given patterns and substitutions.
<code>regularise_version(version[, add_v_prefix])</code>	Regularise given version name by either adding or removing a <i>v</i> affixe.
<code>validate_method(method, valid_methods[, message])</code>	Validate whether given method exists in the given valid methods and returns the method lower cased.
<code>google_sheet_title(url)</code>	Return the title from given <i>Google Sheet</i> url.
<code>slugify(object[, allow_unicode])</code>	Generate a <i>SEO</i> friendly and human-readable slug from given object.

opencolorio_config_aces.utilities.ROOT_BUILD_DEFAULT

```
opencolorio_config_aces.utilities.ROOT_BUILD_DEFAULT =
PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/opencolorio-config-aces/envs/
main/lib/python3.9/site-packages/build')
```

Path subclass for non-Windows systems.

On a POSIX system, instantiating a Path should return this object.

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)`**Methods**

<code>__init__(*args, **kwargs)</code>	
<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>	
<code>keys()</code>	
<code>pop(k[,d])</code>	If key is not found, default is returned if given, otherwise <code>KeyError</code> is raised
<code>popitem()</code>	Remove and return a (key, value) pair as a 2-tuple.
<code>setdefault(key[, default])</code>	Insert key with a value of default if key is not in the dictionary.
<code>update([E,]**F)</code>	If E is present and has a <code>.keys()</code> method, then does: for k in E: D[k] = E[k] If E is present and lacks a <code>.keys()</code> 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]
<code>values()</code>	

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

Return 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)

Return the common ancestor of given iterables.

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)

Return 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**()

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

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)

Convert given vivified data-structure to dictionary.

Parameters

vivified (defaultdict) – Vivified data-structure.

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>)

Print 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_colour_installed

opencolorio_config_aces.utilities.**is_colour_installed**(raise_exception=False)

Return if *Colour* is installed and available.

Parameters

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

Returns

Is *Colour* installed.

Return type

bool

Raises

ImportError – If *Colour* is not installed.

opencolorio_config_aces.utilities.is_jsonpickle_installed

opencolorio_config_aces.utilities.is_jsonpickle_installed(raise_exception=False)

Return if *jsonpickle* is installed and available.

Parameters

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

Returns

Is *jsonpickle* installed.

Return type

bool

Raises

ImportError – If *jsonpickle* is not installed.

opencolorio_config_aces.utilities.is_networkx_installed

opencolorio_config_aces.utilities.is_networkx_installed(raise_exception=False)

Return if *NetworkX* is installed and available.

Parameters

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

Returns

Is *NetworkX* installed.

Return type

bool

Raises

ImportError – If *NetworkX* is not installed.

opencolorio_config_aces.utilities.REQUIREMENTS_TO_CALLABLE

opencolorio_config_aces.utilities.REQUIREMENTS_TO_CALLABLE = {'Colour': <function is_colour_installed>, 'NetworkX': <function is_networkx_installed>, 'jsonpickle': <function is_jsonpickle_installed>}

Mapping of requirements to their respective callables.

_REQUIREMENTS_TO_CALLABLE

[CaseInsensitiveMapping] {'Colour', 'jsonpickle', 'NetworkX', 'OpenImageIO'}

opencolorio_config_aces.utilities.required

opencolorio_config_aces.utilities.required(*requirements)

Decorate a function to check whether various ancillary package requirements are satisfied.

Parameters

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

Return type

object

opencolorio_config_aces.utilities.is_string

`opencolorio_config_aces.utilities.is_string(a)`

Return 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)`

Return 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()`

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

Returns

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

opencolorio_config_aces.utilities.matrix_3x3_to_4x4`opencolorio_config_aces.utilities.matrix_3x3_to_4x4(M)`

Convert given 3x3 matrix *M* to a raveled 4x4 matrix.

Parameters

M (array_like) – 3x3 matrix *M* to convert.

Returns

Raveled 4x4 matrix.

Return type

`list`

opencolorio_config_aces.utilities.multi_replace`opencolorio_config_aces.utilities.multi_replace(name, patterns)`

Update given name by applying in succession the given patterns and substitutions.

Parameters

- **name** (unicode) – Name to update.
- **patterns** (dict) – Dictionary of regular expression patterns and substitution to apply onto the name.

Returns

Updated name.

Return type

unicode

Examples

```
>>> multi_replace(  
...     'Canon Luke Skywalker was weak and powerless.',  
...     {'Canon': 'Legends', 'weak': 'strong', '\w+less': 'powerful'})  
'Legends Luke Skywalker was strong and powerful.'
```

opencolorio_config_aces.utilities.regularise_version`opencolorio_config_aces.utilities.regularise_version(version, add_v_prefix=True)`

Regularise given version name by either adding or removing a *v* affixe.

Parameters

- **version** (str) – Version name to regularise.
- **add_v_prefix** (bool, optional) – Whether to add the *v* affixe.

Returns

Regularise version name.

Return type

`str`

Examples

```
>>> regularise_version("0.1.0")
'v0.1.0'
>>> regularise_version("v0.1.0")
'v0.1.0'
>>> regularise_version("v0.1.0", False)
'0.1.0'
>>> regularise_version("0.1.0", False)
'0.1.0'
```

opencolorio_config_aces.utilities.validate_method

`opencolorio_config_aces.utilities.validate_method(method, valid_methods, message="{0}"
method is invalid, it must be one of {1}!')`

Validate whether given method exists in the given valid methods and returns the method lower cased.

Parameters

- **method** (`str`) – Method to validate.
- **valid_methods** (`Union[Sequence, Mapping]`) – Valid methods.
- **message** (`str`) – Message for the exception.

Returns

Method lower cased.

Return type

`str`

Raises

ValueError – If the method does not exist.

Examples

```
>>> validate_method('Valid', ['Valid', 'Yes', 'Ok'])
'valid'
```

opencolorio_config_aces.utilities.google_sheet_title

`opencolorio_config_aces.utilities.google_sheet_title(url)`

Return the title from given *Google Sheet* url.

Parameters

url (`str`) – *Google Sheet* url to return the title of.

Returns

Google Sheet title.

Return type

`str`

Examples

```
>>> url = (  
...     "https://docs.google.com/spreadsheets/d/"  
...     "1SXpt-USy3HlV2G2qAvh9zit6ZCIND0lfKT07yXJdWLg/"  
...     "export?format=csv&gid=273921464"  
... )  
>>> google_sheet_title(url)  
'OpenColorIO-Config-ACES "Reference" Transforms - v...'
```

opencolorio_config_aces.utilities.slugify

opencolorio_config_aces.utilities.**slugify**(object_, allow_unicode=False)

Generate a *SEO* friendly and human-readable slug from given object.

Convert to ASCII if *allow_unicode* is *False*. Convert spaces or repeated dashes to single dashes. Remove characters that aren't alphanumerics, underscores, or hyphens. Convert to lowercase. Also strip leading and trailing whitespace, dashes, and underscores.

Parameters

- **object** (*object*) – Object to convert to a slug.
- **allow_unicode** (*bool*) – Whether to allow unicode characters in the generated slug.

Returns

Generated slug.

Return type

str

References

- <https://github.com/django/django/blob/0dd29209091280ccf34e07c9468746c396b7778e/django/utils/text.py#L400>

Examples

```
>>> slugify(  
...     " Jack & Jill like numbers 1,2,3 and 4 and silly characters ?%.$!/"  
... )  
'jack-jill-like-numbers-123-and-4-and-silly-characters'
```

3.1.2 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.ColorspaceDescriptionStyle method), 38

`__init__()` (opencolorio_config_aces.ConfigData method), 23

`__init__()` (opencolorio_config_aces.ProfileVersion method), 19

`__init__()` (opencolorio_config_aces.utilities.DocstringDict method), 44

A

`active_displays` (opencolorio_config_aces.ConfigData attribute), 22

`active_views` (opencolorio_config_aces.ConfigData attribute), 22

B

`build_aces_conversion_graph()` (in module opencolorio_config_aces), 35

`BUILTIN_TRANSFORMS` (in module opencolorio_config_aces), 26

C

`classify_aces_ctl_transforms()` (in module opencolorio_config_aces), 32

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

`colorspace_factory()` (in module opencolorio_config_aces), 28

`ColorspaceDescriptionStyle` (class in opencolorio_config_aces), 38

`colorspaces` (opencolorio_config_aces.ConfigData attribute), 22

`common_ancestor()` (in module opencolorio_config_aces.utilities), 45

`ConfigData` (class in opencolorio_config_aces), 20

`conversion_path()` (in module opencolorio_config_aces), 35

`ctl_transform_to_node()` (in module opencolorio_config_aces), 36

D

`default_view_transform` (opencolorio_config_aces.ConfigData attribute), 23

`description` (opencolorio_config_aces.ConfigData attribute), 22

`deserialize_config_data()` (in module opencolorio_config_aces), 24

`discover_aces_ctl_transforms()` (in module opencolorio_config_aces), 33

`discover_clf_transforms()` (in module opencolorio_config_aces), 12

`DocstringDict` (class in opencolorio_config_aces.utilities), 44

F

`file_rules` (opencolorio_config_aces.ConfigData attribute), 22

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

`filter_ctl_transforms()` (in module opencolorio_config_aces), 33

`filter_nodes()` (in module opencolorio_config_aces), 36

`first_item()` (in module opencolorio_config_aces.utilities), 44

G

`generate_clf_transform()` (in module opencolorio_config_aces), 14

`generate_clf_transforms_arri()` (in module opencolorio_config_aces.clf), 15

`generate_clf_transforms_bmdfilm()` (in module opencolorio_config_aces.clf), 15

`generate_clf_transforms_canon()` (in module opencolorio_config_aces.clf), 16

`generate_clf_transforms_davinci()` (in module opencolorio_config_aces.clf), 16

`generate_clf_transforms_itu()` (in module opencolorio_config_aces.clf), 17

`generate_clf_transforms_ocio()` (in module opencolorio_config_aces.clf), 17

`generate_clf_transforms_panasonic()` (in module opencolorio_config_aces.clf), 17

`generate_clf_transforms_red()` (in module opencolorio_config_aces.clf), 17

`generate_clf_transforms_sony()` (in module opencolorio_config_aces.clf), 18

`generate_config()` (in module `opencolorio_config_aces`), [25](#)
`generate_config_aces()` (in module `opencolorio_config_aces`), [39](#)
`generate_config_cg()` (in module `opencolorio_config_aces`), [40](#)
`generate_config_studio()` (in module `opencolorio_config_aces`), [41](#)
`git_describe()` (in module `opencolorio_config_aces.utilities`), [49](#)
`google_sheet_title()` (in module `opencolorio_config_aces.utilities`), [51](#)
`group_transform_factory()` (in module `opencolorio_config_aces`), [29](#)

I

`inactive_colorspaces` (`opencolorio_config_aces.ConfigData` attribute), [23](#)
`is_colour_installed()` (in module `opencolorio_config_aces.utilities`), [47](#)
`is_iterable()` (in module `opencolorio_config_aces.utilities`), [49](#)
`is_jsonpickle_installed()` (in module `opencolorio_config_aces.utilities`), [48](#)
`is_networkx_installed()` (in module `opencolorio_config_aces.utilities`), [48](#)
`is_string()` (in module `opencolorio_config_aces.utilities`), [49](#)

L

`look_factory()` (in module `opencolorio_config_aces`), [29](#)
`looks` (`opencolorio_config_aces.ConfigData` attribute), [22](#)

M

`major` (in module `opencolorio_config_aces`), [19](#)
`major` (`opencolorio_config_aces.ProfileVersion` attribute), [18](#)
`matrix_3x3_to_4x4()` (in module `opencolorio_config_aces.utilities`), [50](#)
`message_box()` (in module `opencolorio_config_aces.utilities`), [46](#)
`minor` (in module `opencolorio_config_aces`), [19](#)
`minor` (`opencolorio_config_aces.ProfileVersion` attribute), [18](#)
`multi_replace()` (in module `opencolorio_config_aces.utilities`), [50](#)

N

`name` (`opencolorio_config_aces.ConfigData` attribute), [21](#)
`named_transform_factory()` (in module `opencolorio_config_aces`), [29](#)
`named_transforms` (`opencolorio_config_aces.ConfigData` attribute), [22](#)

`node_to_ctl_transform()` (in module `opencolorio_config_aces`), [37](#)

P

`paths_common_ancestor()` (in module `opencolorio_config_aces.utilities`), [45](#)
`plot_aces_conversion_graph()` (in module `opencolorio_config_aces`), [37](#)
`print_aces_taxonomy()` (in module `opencolorio_config_aces`), [34](#)
`print_clf_taxonomy()` (in module `opencolorio_config_aces`), [13](#)
`produce_transform()` (in module `opencolorio_config_aces`), [30](#)
`profile_version` (`opencolorio_config_aces.ConfigData` attribute), [21](#)
`PROFILE_VERSION_DEFAULT` (in module `opencolorio_config_aces`), [19](#)
`ProfileVersion` (class in `opencolorio_config_aces`), [18](#)

R

`regularise_version()` (in module `opencolorio_config_aces.utilities`), [50](#)
`required()` (in module `opencolorio_config_aces.utilities`), [48](#)
`REQUIREMENTS_TO_CALLABLE` (in module `opencolorio_config_aces.utilities`), [48](#)
`roles` (`opencolorio_config_aces.ConfigData` attribute), [22](#)
`ROOT_BUILD_DEFAULT` (in module `opencolorio_config_aces.utilities`), [43](#)

S

`schema_version` (`opencolorio_config_aces.ConfigData` attribute), [21](#)
`search_path` (`opencolorio_config_aces.ConfigData` attribute), [22](#)
`serialize_config_data()` (in module `opencolorio_config_aces`), [25](#)
`shared_views` (`opencolorio_config_aces.ConfigData` attribute), [22](#)
`slugify()` (in module `opencolorio_config_aces.utilities`), [52](#)
`SUPPORTED_PROFILE_VERSIONS` (in module `opencolorio_config_aces`), [20](#)

T

`TRANSFORM_FACTORIES` (in module `opencolorio_config_aces`), [28](#)
`transform_factory()` (in module `opencolorio_config_aces`), [30](#)

U

`unclassify_clf_transforms()` (in module `opencolorio_config_aces`), [14](#)

`unclassify_ctl_transforms()` (in module `opencolorio_config_aces`), [34](#)

V

`validate_config()` (in module `opencolorio_config_aces`), [25](#)

`validate_method()` (in module `opencolorio_config_aces.utilities`), [51](#)

`version_aces_dev()` (in module `opencolorio_config_aces`), [32](#)

`version_config_mapping_file()` (in module `opencolorio_config_aces`), [38](#)

`view_transform_factory()` (in module `opencolorio_config_aces`), [31](#)

`view_transforms` (`opencolorio_config_aces.ConfigData` attribute), [22](#)

`viewing_rules` (`opencolorio_config_aces.ConfigData` attribute), [22](#)

`views` (`opencolorio_config_aces.ConfigData` attribute), [22](#)

`vivification()` (in module `opencolorio_config_aces.utilities`), [45](#)

`vivified_to_dict()` (in module `opencolorio_config_aces.utilities`), [46](#)