OpenColorIO Configuration for ACES Documentation

Release 0.3.1

OpenColorIO Contributors

Oct 07, 2022

CONTENTS

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.

CHAPTER

ONE

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.
- Generators producing the *OCIO* **CG** and **Studio** configurations.
- Included *CLF* transforms along with generator and discovery support.

CHAPTER

TWO

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	*invoke build-config-reference*
Build (Docker)	*invoke docker-run-build-config-reference*
Updating Mapping File	*invoke update-mapping-file-reference*

CG Config

Task	Command
Build	*invoke build-config-cg*
Build (Docker)	*invoke docker-run-build-config-cg*
Updating Mapping File	*invoke update-mapping-file-cg*

Reference Config

Task	Command
Build	*invoke build-config-studio*
Build (Docker)	*invoke docker-run-build-config-studio*
Updating Mapping File	*invoke update-mapping-file-studio*

CHAPTER

THREE

1.3 API REFERENCE

3.1 API Reference

3.1.1 OpenColorIO Configuration for ACES

Common LUT Format Discovery

opencolorio_config_aces

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

$$\{\text{“}family''_1 : \{\text{“}genus''_1 : \{\}_{CLF_1}, \ldots, \text{“}family''_n : \{\text{“}genus''_2 : \{\}_{CLF_2}\}\}$$

where

$$\{\}_{CLF_n} = \{\text{“}basename''_n : CLFTransform_n, \ldots, \text{“}basename''_{n+1} : CLFTransform_{n+1}\}$$

Return type

dict

Examples

```
>>> clf_transforms = classify_clf_transforms(
...     discover_clf_transforms())
>>> family = sorted(clf_transforms.keys())[0]
>>> str(family)
'blackmagic'
>>> genera = sorted(clf_transforms[family])
>>> print(genera)
['Input']
>>> genus = genera[0]
>>> sorted(clf_transforms[family][genus].items())[:2]
[('BlackmagicDesign.Input.BMDFilm_Gen5_Log-Curve', CLFTransform('blackmagic...input..
↪.BlackmagicDesign.Input.BMDFilm_Gen5_Log-Curve.clf')), ('BlackmagicDesign.Input.
↪BMDFilm_WideGamut_Gen5_to_ACES2065-1', CLFTransform('blackmagic...input...
↪BlackmagicDesign.Input.BMDFilm_WideGamut_Gen5_to_ACES2065-1.clf'))]
```

opencolorio_config_aces.discover_clf_transforms

opencolorio_config_aces.**discover_clf_transforms**(*root_directory='/home/docs/checkouts/readthedocs.org/user_build_config-aces/envs/v0.3.1/lib/python3.8/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

$$\{\text{“}directory_1\text{”} : [\text{“}transform_a.clf\text{”}, \text{“}transform_b.clf\text{”}], \ldots, \text{“}directory_n\text{”} : [\text{“}transform_c.clf\text{”}, \text{“}transform_d.clf\text{”}]\}$$

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]
['BlackmagicDesign.Input.BMDFilm_Gen5_Log-Curve.clf', 'BlackmagicDesign.Input.
↪BMDFilm_WideGamut_Gen5_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

$$[CLFTransform_1, \ldots, CLFTransform_n]$$

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

$$[CLFTransform_1, \ldots, CLFTransform_n]$$

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('blackmagic...input...BlackmagicDesign.Input.BMDFilm_Gen5_Log-Curve.clf
↪')
```

Common LUT Format Generation

opencolorio_config_aces

generate_clf_transform(filename, transforms, ...)	Take a list of transforms and some metadata and write a *CLF* 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`

generate_clf_transforms_bmdfilm(output_directory)	Make the CLF file for BMDFilm_WideGamut_Gen5 plus matrix/curve CLFs.
generate_clf_transforms_davinci(output_directory)	Make the CLF file for DaVinci Intermediate Wide Gamut plus matrix/curve CLFs.
generate_clf_transforms_itu(output_directory)	Generate OCIO Utility CLF transforms.
generate_clf_transforms_ocio(output_directory)	Generate OCIO Utility CLF transforms.
generate_clf_transforms_panasonic(...)	Make the CLF file for V-Log - V-Gamut plus matrix/curve CLFs.
generate_clf_transforms_red(output_directory)	Make the CLF file for RED Log3G10 REDWideGamutRGB plus matrix/curve CLFs.

opencolorio_config_aces.clf.generate_clf_transforms_bmdfilm

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

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

Returns
Dictionary 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_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*.

Generation

Config Generation Common Objects

`opencolorio_config_aces`

`ConfigData`(schema_version, profile_version, ...)	Define the data container for an *OpenColorIO* config.
`VersionData`([major, minor])	Define the data container for a two component version identifier.
`deserialize_config_data`(path)	Deserialize the *JSON OpenColorIO* config data container at given path.
`generate_config`(data[, config_name, ...])	Generate the *OpenColorIO* config from given data.
`serialize_config_data`(data, path)	Serialize the *OpenColorIO* config data container as a *JSON* file.
`validate_config`(config)	Validate given *OpenColorIO* config.

opencolorio_config_aces.ConfigData

class opencolorio_config_aces.**ConfigData**(*schema_version: ~opencolorio_config_aces.config.generation.common.VersionData = VersionData(major=1, minor=0), profile_version: ~opencolorio_config_aces.config.generation.common.VersionData = VersionData(major=2, minor=0), 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** (`VersionData`, `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.NamedTransfom* 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.common.VersionData

profile_version

Type
opencolorio_config_aces.config.generation.common.VersionData

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.common.VersionData = VersionData(major=1, minor=0), profile_version: ~opencolorio_config_aces.config.generation.common.VersionData = VersionData(major=2, minor=0), 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
profile_version
schema_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.VersionData

class opencolorio_config_aces.**VersionData**(*major: int = 1*, *minor: int = 0*)

Define the data container for a two component version identifier.

Parameters

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

major

Type
int

minor

Type
int

__init__(*major: int = 1*, *minor: int = 0*) → None

Methods

`__init__`([major, minor])	

Attributes

`major`	
`minor`	

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

TRANSFORM_FACTORIES	dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's (key, value) pairs dict(iterable) -> new dictionary initialized as if via: d = {} for k, v in iterable: d[k] = v dict(**kwargs) -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2).
colorspace_factory(name[, family, encoding, ...])	*OpenColorIO Colorspace* factory.
group_transform_factory(transforms)	*OpenColorIO GroupTransform* factory.
look_factory(name[, process_space, ...])	*OpenColorIO Look* factory.
named_transform_factory(name[, family, ...])	*OpenColorIO NamedTransform* factory.
produce_transform(transform)	Produce given transform.
transform_factory(**kwargs)	*OpenColorIO* transform factory.
view_transform_factory(name[, family, ...])	*OpenColorIO ViewTransform* factory.

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>}**

dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via:
d = {} for k, v in iterable:

d[k] = v

dict(kwargs) -> new dictionary initialized with the name=value pairs**
in the keyword argument list. For example: dict(one=1, two=2)

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

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

$$\{\text{“}family''_1 : \{\text{“}genus''_1 : \{\}_{CTL_1}, \ldots, \text{“}family''_n : \{\text{“}genus''_2 : \{\}_{CTL_2}\}\}$$

where

$$\{\}_{CTL_n} = \{\text{“}basename''_n : CTLTransform_n, \ldots, \text{“}basename''_{n+1} : CTLTransform_{n+1}\}$$

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_config-aces/envs/v0.3.1/lib/python3.8/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

$$\{\text{“}directory_1\text{''} : [\text{“}transform_a.ctl\text{''}, \text{“}transform_b.ctl\text{''}],\ \ldots,\ \text{“}directory_n\text{''} : [\text{“}transform_c.ctl\text{''}, \text{“}transform_d.ctl\text{''}]\}$$

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

$$[CTLTransform_1, \ldots, CTLTransform_n]$$

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

$$[CTLTransform_1, \ldots, CTLTransform_n]$$

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

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

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_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 pyraphviz.

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

ColorspaceDescriptionStyle(value)	Enum storing the various *OpenColorIO Colorspace* description styles.
version_config_mapping_file([path])	Return the current version of given *CSV* mapping file.
generate_config_aces([config_name, ...])	Generate the *aces-dev* reference implementation *OpenColorIO* config using the *Mapping* 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_ config-aces/envs/v0.3.1/lib/python3.8/site-packages/opencolorio_config_aces/config/reference/generate/r Config-ACES Reference Transforms - v0.2.0 - 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*, *validate=True*, *describe=ColorspaceDescriptionStyle.SHORT_UNION*, *config_mapping_file_path=PosixPath('/home/docs/checkouts/readthed config-aces/envs/v0.3.1/lib/python3.8/site-packages/opencolorio_config_aces/config/reference/generate/resources/ Config-ACES Reference Transforms - v0.2.0 - 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.
- **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

generate_config_cg([data, config_name, ...])	Generate the *ACES* Computer Graphics (CG) *OpenColorIO* config.

opencolorio_config_aces.generate_config_cg

opencolorio_config_aces.**generate_config_cg**(*data=None*, *config_name=None*, *validate=True*, *describe=ColorspaceDescriptionStyle.SHORT_UNION*, *config_mapping_file_path=PosixPath('/home/docs/checkouts/readthedoc config-aces/envs/v0.3.1/lib/python3.8/site-packages/opencolorio_config_aces/config/cg/generate/resources/OpenCol Config-ACES CG Transforms - v0.2.0 - 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.
- **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*, *validate=True*, *describe=ColorspaceDescriptionStyle.SHORT_UNION*, *config_mapping_file_path=PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/opencolorio-config-aces/envs/v0.3.1/lib/python3.8/site-packages/opencolorio_config_aces/config/studio/generate/resources/Config-ACES Studio Transforms - v0.1.0 - 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.
- **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

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

`__init__`(*args, **kwargs)	
`clear`()	
`copy`()	
`fromkeys`([value])	Create a new dictionary with keys from iterable and values set to value.
`get`(key[, default])	Return the value for key if key is in the dictionary, else default.
`items`()	
`keys`()	
`pop`(k[,d])	If key is not found, d is returned if given, otherwise KeyError is raised
`popitem`()	Remove and return a (key, value) pair as a 2-tuple.
`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*)

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-USy3HlV2G2qAvh9zit6ZCINDOlfKT07yXJdWLg/"
...     "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

:cite:`DjangoSoftwareFoundation2022`

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

CHAPTER

FOUR

1.4 ABOUT

OpenColorIO Configuration for ACES by OpenColorIO Contributors

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

INDEX

Symbols

A

B

C

D

F

G