python-cwlgen Documentation

Release 0.1

Kenzo-Hugo Hillion, Hervé Ménager

Jun 09, 2020

Contents

Warning: `python-cwlgen` is now deprecated, please use cwl-utils >= 0.4.

Example migration:

```
from cwl_utils import parser_v1_0

# You could alias this as cwlgen to simplify the migration
from cwl_utils import parser_v1_0 as cwlgen
```

CHAPTER 1

Migration notes:

- Method changes
  - `get_dict()` → `save()`
  - `parse_cwl(cwlfile)` → `load_document(cwlfile)`
  - `parse_dict` → No super clear analogue, but loaded through `_RecordLoader(CommandLineTool)` || `_UnionLoader((CommandLineToolLoader, ...workflow + other loaders)`
- Field names:
  - Uses `camelCase` instead of `snake_case`
  - No more special field names, eg: - `tool_id` | `workflow_id` | `input_id` | etc → `id` - `StepInput`: `inputs` → `in_`
- Other notes:
  - Classes aren't nested anymore, ie: `cwlgen.InitialWorkDirRequirement.Dirent` → `cwlutils`.
  - Take care if you're migrating to a newer spec, as some classes might have changed names (notably: `InputParameter` -> `WorkflowInputParameter`)
  - Don't forget to catch all references of cwlgen, as missing one (or using mismatch versions of the parser) will cause:

```
raise RepresenterError('cannot represent an object: %s' %␣
↪(data,))
ruamel.yaml.representer.RepresenterError: cannot represent an␣
↪object:
<cwlgen.common.CommandInputArraySchema object at 0x1100a5780>
```

If you have issues with the migration, please see this thread or raise an issue on CWLUtils.

Python-cwlgen is a python library for the programmatic generation of CWL v1.0. It supports the generation of CommandLineTool and Workflows.

The library works for both Python 2.7.12+ and 3.6.0+.

CHAPTER 2

Quick-start

You can install Python-CWLGen through pip with the following command:

```
pip install cwlgen
```

The classes very closely (if not exactly) mirror the CWL v1.0 specification. You can find more about their parameters in the following specifications:

- *cwlgen.CommandLineTool*
- cwlgen.Workflow

CHAPTER 3

Python-cwlgen

3.1 Installation

Note: We highly recommend the use of a virtual environment with Python 3.6.0 using virtualenv or conda.

3.1.1 python-cwlgen dependencies

python-cwlgen has been primarily tested using Python3 and uses the following libraries:

- `ruamel.yaml` (between 0.12.4 and 0.15.87)
- `six` (1.10.0)

The project has been designed to work with Python 2.7+ and has accompanying tests, however please raise an issue if you have incompatibility issues.

3.1.2 Installation procedure

Pip

You can use pip to install the latest version from pypi:

```
pip install cwlgen
```

Manually

Clone the repository and install cwlgen with the following command:

```
git clone https://github.com/common-workflow-language/python-cwlgen.git
cd python-cwlgen
pip install .
```

3.1.3 Uninstallation procedure

Pip

You can remove python-cwlgen with the following command:

```
pip uninstall cwlgen
```

Note: This will not uninstall dependencies. To do so you can make use of pip-autoremove.

3.2 User Guide

This user guide assumes you have at least some basic knowledge about CWL.

Note: Here is a CWL user guide for an introduction to tool and workflows wrappers.

The aim is to help you through the different steps to build your CWL tool with python-cwlgen.

Note: If you find a bug, have any questions or suggestions, please submit an issue on Github.

3.2.1 Basic example

Through this little tutorial, we will go step by step through the example you can find on Github. It aims to wrap the *grep* command.

Initialize your tool

You need to initialize a *CommandLineTool* object

```
import cwlgen
cwl_tool = cwlgen.CommandLineTool(tool_id='grep',
                                  label='print lines matching a pattern
 ↪',
                                  base_command='grep')
```

Now that you have your object, you can attach the different elements of a tool description.

Add Inputs

Now we need to add inputs to our tool. We are going to only wrap a simple version of the *grep* command with a input file and a pattern.

First the input file:

```
file_binding = cwlgen.CommandLineBinding(position=2)
input_file = cwlgen.CommandInputParameter('input_file',
                                          param_type='File',
                                          input_binding=file_binding,
                                          doc='input file from which␣
 ↪you want to look for the pattern')
cwl_tool.inputs.append(input_file)
```

And finally the pattern:

```
pattern_binding = cwlgen.CommandLineBinding(position=1)
pattern = cwlgen.CommandInputParameter('pattern',
                                       param_type='string',
                                       input_binding=pattern_binding,
                                       doc='pattern to find in the␣
 ↪input file')
cwl_tool.inputs.append(pattern)
```

Note: You can specify more information concerning your inputs: Input documentation

This is it for the inputs, now let's add some outputs and the description will be ready to be tested.

Add an Output

The only output which is retrieved in our example is a File with the line containing the pattern. Here is how to add this output:

```
output = cwlgen.CommandOutputParameter('output',
                                       param_type='stdout',
                                       doc='lines found with the␣
↪pattern')
cwl_tool.outputs.append(output)
# Now specify a name for your output file
cwl_tool.stdout = "grep.txt"
```

Add Documentation and Metadata

You can ask bunch of information and metadata concerning your tool. For instance you can add some documentation:

```
cwl_tool.doc = "grep searches for a pattern in a file."
```

For the metadata:

```
metadata = {'name': 'grep',
            'about' : 'grep searches for a pattern in a file.'}
cwl_tool.metadata = cwlgen.Metadata(**metadata)
```

Write your tool

Finally, you can export your tool description with the *export()* method.

```
cwl_tool.export()  # On STDOUT
cwl_tool.export(outfile="grep.cwl")  # As a file (grep.cwl)
```

You can then try your tool description (using cwltool for instance):

```
cwltool grep.cwl --input_file underdog_lyrics.txt --pattern lost
```

3.3 References

3.3.1 Common Workflow Language

CWL is developed by an informal, multi-vendor working group consisting of organizations and individuals aiming to enable scientists to share data analysis workflows. The CWL project is on Github.

CHAPTER 4

Python-cwlgen API documentation

4.1 API classes

4.1.1 Workflow and CommandLineTool

See the links below to the *CommandLineTool* and *Workflow* classes:

- *cwlgen.CommandLineTool*
- cwlgen.Workflow

4.1.2 Requirements

Requirement

This is the (abstract) base requirement class.

class cwlgen.**Requirement**(*req_class*)

__hash__()
Return hash(self).

__init__(*req_class*)

Parameters **req_class** (*STRING*) – requirement class

ignore_fields_on_parse = ['class']
Requirement that must be met in order to execute the process.

InlineJavascriptRequirement

class cwlgen.**InlineJavascriptRequirement**(*expression_lib=None*)
Indicates that the workflow platform must support inline Javascript expressions. If this requirement is not present, the workflow platform must not perform expression interpolatation.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#InlineJavascriptRequirement

__init__(*expression_lib=None*)

Parameters **expression_lib** (*list[STRING]*) – List of Strings

SchemaDefRequirement

See the Schema section Below:

- *cwlgen.SchemaDefRequirement*

SubworkflowFeatureRequirement

class cwlgen.**SubworkflowFeatureRequirement**
Indicates that the workflow platform must support nested workflows in the run field of WorkflowStep.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#SubworkflowFeatureRequirement

__init__()

Parameters **req_class** (*STRING*) – requirement class

ScatterFeatureRequirement

class cwlgen.**ScatterFeatureRequirement**
Indicates that the workflow platform must support the scatter and scatterMethod fields of WorkflowStep.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#ScatterFeatureRequirement

__init__()

Parameters **req_class** (*STRING*) – requirement class

MultipleInputFeatureRequirement

class cwlgen.**MultipleInputFeatureRequirement**

Indicates that the workflow platform must support multiple inbound data links listed in the source field of WorkflowStepInput.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#MultipleInputFeatureRequirement

__init__()

Parameters **req_class** (*STRING*) – requirement class

StepInputExpressionRequirement

class cwlgen.**StepInputExpressionRequirement**

Indicate that the workflow platform must support the valueFrom field of WorkflowStepInput.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#StepInputExpressionRequirement

__init__()

Parameters **req_class** (*STRING*) – requirement class

DockerRequirement

class cwlgen.**DockerRequirement**(*docker_pull=None*, *docker_load=None*, *docker_file=None*, *docker_import=None*, *docker_image_id=None*, *docker_output_dir=None*)

Indicates that a workflow component should be run in a Docker container, and specifies how to fetch or build the image.

Documentation: https://www.commonwl.org/v1.0/CommandLineTool.html#DockerRequirement

__init__(*docker_pull=None*, *docker_load=None*, *docker_file=None*, *docker_import=None*, *docker_image_id=None*, *docker_output_dir=None*)

Parameters

- **docker_pull** (*STRING*) – image to retrive with docker pull
- **docker_load** (*STRING*) – HTTP URL from which to download Docker image
- **docker_file** (*STRING*) – supply the contents of a Dockerfile

- **docker_import** (*STRING*) – HTTP URL to download and gunzip a Docker images
- **docker_image_id** (*STRING*) – Image id for docker run
- **docker_output_dir** (*STRING*) – designated output dir inside the Docker container

SoftwareRequirement

class cwlgen.**SoftwareRequirement**(*packages=None*)

A list of software packages that should be configured in the environment of the defined process.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#SoftwareRequirement

__init__(*packages=None*)

Parameters **req_class** (*STRING*) – requirement class

class cwlgen.SoftwareRequirement.**SoftwarePackage**(*package*, *version=None*, *specs=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#SoftwarePackage

__init__(*package*, *version=None*, *specs=None*)

Parameters

- **package** – The name of the software to be made available. If the name is common, inconsistent, or otherwise ambiguous it should be combined with one or more identifiers in the specs field
- **version** – The (optional) versions of the software that are known to be compatible.
- **specs** – One or more IRIs identifying resources for installing or enabling the software in 'package'

InitialWorkDirRequirement

class cwlgen.**InitialWorkDirRequirement**(*listing*)

Define a list of files and subdirectories that must be created by the workflow platform in the designated output directory prior to executing the command line tool.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#InitialWorkDirRequirement

__init__(*listing*)

Parameters **listing** (*array<File | Directory | Dirent | string | Expression> | string | Expression*) – The list of files or subdirectories that must be placed in the designated output directory prior to executing the command line tool.

class cwlgen.InitialWorkDirRequirement.**Dirent**(*entry*, *entryname=None*, *writable=None*)

Define a file or subdirectory that must be placed in the designated output directory prior to executing the command line tool. May be the result of executing an expression, such as building a configuration file from a template.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#Dirent

__init__(*entry*, *entryname=None*, *writable=None*)

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

EnvVarRequirement

class cwlgen.**EnvVarRequirement**(*env_def*)

Define a list of environment variables which will be set in the execution environment of the tool. See EnvironmentDef for details.

Documentation: https://www.commonwl.org/v1.0/CommandLineTool.html#EnvVarRequirement

__init__(*env_def*)

Parameters **env_def** (*list[EnvironmentDef]*) – The list of environment variables.

class cwlgen.EnvVarRequirement.**EnvironmentDef**(*env_name*, *env_value*)

Define an environment variable that will be set in the runtime environment by the workflow platform when executing the command line tool. May be the result of executing an expression, such as getting a parameter from input.

Documentation: https://www.commonwl.org/v1.0/CommandLineTool.html#EnvironmentDef

__init__(*env_name*, *env_value*)

Parameters

- **env_name** (*STRING*) – The environment variable name
- **env_value** (*STRING*) – The environment variable value

ShellCommandRequirement

class cwlgen.**ShellCommandRequirement**

Modify the behavior of CommandLineTool to generate a single string containing a shell command line.

Documentation: https://www.commonwl.org/v1.0/CommandLineTool.html#ShellCommandRequirement

__init__()

Parameters **req_class** (*STRING*) – requirement class

ResourceRequirement

class cwlgen.**ResourceRequirement**(*cores_min=None*, *cores_max=None*, *ram_min=None*, *ram_max=None*, *tmpdir_min=None*, *tmpdir_max=None*, *outdir_min=None*, *outdir_max=None*)

Specify basic hardware resource requirements.

Documentation: https://www.commonwl.org/v1.0/CommandLineTool.html#ResourceRequirement

__init__(*cores_min=None*, *cores_max=None*, *ram_min=None*, *ram_max=None*, *tmpdir_min=None*, *tmpdir_max=None*, *outdir_min=None*, *outdir_max=None*)

Parameters

- **cores_min** (*string | float | None*) – Minimum reserved number of CPU cores
- **cores_max** (*string | float | None*) – Maximum reserved number of CPU cores
- **ram_min** (*string | float | None*) – Minimum reserved RAM in mebibytes (2**20)
- **ram_max** (*string | float | None*) – Maximum reserved RAM in mebibytes (2**20)
- **tmpdir_min** (*string | float | None*) – Minimum reserved filesystem based storage for the designated temporary directory, in mebibytes (2**20)
- **tmpdir_max** (*string | float | None*) – Maximum reserved filesystem based storage for the designated temporary directory, in mebibytes (2**20)

- **outdir_min** (*string | float | None*) – Minimum reserved filesystem based storage for the designated output directory, in mebibytes (2**20)
- **outdir_max** (*string | float | None*) – Maximum reserved filesystem based storage for the designated output directory, in mebibytes (2**20)

4.1.3 Schema

class cwlgen.**SchemaDefRequirement**(*types*)

This field consists of an array of type definitions which must be used when interpreting the inputs and outputs fields. When a type field contain a IRI, the implementation must check if the type is defined in schemaDefs and use that definition. If the type is not found in schemaDefs, it is an error. The entries in schemaDefs must be processed in the order listed such that later schema definitions may refer to earlier schema definitions.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#SchemaDefRequirement

__init__(*types*)

Parameters **types** (*list[InputRecordSchema | InputEnumSchema | InputArraySchema]*) – The list of type definitions.

Workflow Input Schema

class cwlgen.SchemaDefRequirement.**InputRecordSchema**(*label=None*, *name=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#InputRecordSchema

class InputRecordField(*name*, *type*, *doc=None*, *input_binding=None*, *label=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#InputRecordField

__init__(*name*, *type*, *doc=None*, *input_binding=None*, *label=None*)

Parameters

- **name** –
- **input_type** (*CWLType | InputRecordSchema | InputEnumSchema | InputArraySchema | string | array<CWLType | InputRecordSchema | InputEnumSchema | InputArraySchema | string>*) –
- **doc** – A documentation string for this field
- **input_binding** (CommandLineBinding) –
- **label** –

__init__(*label=None*, *name=None*)

Parameters

- **fields** (*array<InputRecordField>*) – Defines the fields of the record.
- **label** – A short, human-readable label of this object.
- **name** – NF (Name of the InputRecord)

class cwlgen.SchemaDefRequirement.**InputEnumSchema**(*symbols*, *label=None*, *name=None*, *input_binding=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#InputEnumSchema

__init__(*symbols*, *label=None*, *name=None*, *input_binding=None*)

Parameters

- **symbols** (*list[STRING]*) – Defines the set of valid symbols.
- **label** (*STRING*) – A short, human-readable label of this object.
- **name** (*STRING*) –
- **input_binding** (CommandLineBinding) –

class cwlgen.SchemaDefRequirement.**InputArraySchema**(*items*, *label=None*, *input_binding=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#InputArraySchema

__init__(*items*, *label=None*, *input_binding=None*)

Parameters

- **items** (*CWLType | InputRecordSchema | InputEnumSchema | InputArraySchema | string | array<CWLType | InputRecordSchema | InputEnumSchema | InputArraySchema | string>*) – Defines the type of the array elements.
- **label** (*STRING*) – A short, human-readable label of this object.
- **input_binding** (CommandLineBinding) –

- CommandLineBinding

  When listed under inputBinding in the input schema, the term "value" refers to the the corresponding value in the input object. For binding objects listed in CommandLineTool.arguments, the term "value" refers to the effective value after evaluating valueFrom.

4.1.4 Import CWL

As of release v0.3.0 the existing importing CWL has been replaced by an automated deserialization. Each function that inherits from the `Serializable` class will have a `parse_dict` method.

If you're adding a class and want to provide a hint on how to parse a particular field, you can add a static `parse_types` dictionary onto your class with the fieldname and a list of types that you want to try and parse as. If your input can be a list (eg: `T[]`), or a dictionary with the identifier as the key (eg: `{ $identifier: T }`, you can let your type be `[T]` in the `parse_types` dict. It will automatically inject this identifier in the constructor. See the `Serializable.parse_dict` class for more information.

```
class Workflow:
    parse_types = {
        # Parse inputs as : [InputParameter] or { id: InputParameter }
        "inputs": [[InputParameter]],

        # will attempt to parse extraParam as a string, then␣
↪SecondaryType,
        # then (TertiaryType[] || { $identifier: TertiaryType }
        "extraParam": [str, SecondaryType, [TertiaryType]]
    }
```

cwlgen.**parse_cwl**(*cwl_path*)

Method that parses a CWL file and will a `cwlgen.Workflow` or *cwlgen.CommandLineTool*. Note: this will not import additional files.

Parameters **cwl_path** (*str*) – PATH to the CWL file

Returns `cwlgen.Workflow` | *cwlgen.CommandLineTool*

cwlgen.**parse_cwl_dict**(*cwl_dict*)

Method that parses a dictionary and will return a `cwlgen.Workflow` or *cwlgen.CommandLineTool*.

Parameters **cwl_dict** (`dict`) – The dictionary to pass, must contain a 'class' field.

Returns `cwlgen.Workflow` | *cwlgen.CommandLineTool*

4.2 CommandLineTool API classes

4.2.1 CommandLineTool

class cwlgen.**CommandLineTool**(*tool_id=None*, *base_command=None*, *label=None*, *doc=None*, *cwl_version='v1.0'*, *stdin=None*, *stderr=None*, *stdout=None*, *path=None*, *requirements=None*, *hints=None*, *inputs=None*, *outputs=None*, *arguments=None*)

Contain all informations to describe a CWL command line tool.

__init__(*tool_id=None*, *base_command=None*, *label=None*, *doc=None*, *cwl_version='v1.0'*, *stdin=None*, *stderr=None*, *stdout=None*, *path=None*, *requirements=None*, *hints=None*, *inputs=None*, *outputs=None*, *arguments=None*)

Parameters

- **tool_id** (*str*) – Unique identifier for this tool
- **base_command** (*str | list[STRING]*) – command line for the tool
- **label** (*str*) – label of this tool
- **doc** (*str*) – documentation for the tool, usually longer than the label
- **cwl_version** (*str*) – version of the CWL tool
- **stdin** (*str*) – path to a file whose contents must be piped into stdin
- **stderr** (*str*) – capture stderr into the given file
- **stdout** (*str*) – capture stdout into the given file

inputs (*cwlgen.CommandInputParameter* objects), outputs (*cwlgen.CommandOutputParameter* objects), arguments (*cwlgen.CommandLineBinding* objects), hints (any | *cwlgen.Requirement* objects) and requirements (*cwlgen.Requirement* objects) are stored in lists which are initialized empty.

export(*outfile=None*)

Export the tool in CWL either on STDOUT or in outfile.

4.2.2 Input and outputs

CommandInputParameter

class cwlgen.**CommandInputParameter**(*param_id*, *label=None*, *secondary_files=None*, *param_format=None*, *streamable=None*, *doc=None*, *input_binding=None*, *default=None*, *param_type=None*)

An input parameter for a *cwlgen.CommandLineTool*.

__init__(*param_id*, *label=None*, *secondary_files=None*, *param_format=None*, *streamable=None*, *doc=None*, *input_binding=None*, *default=None*, *param_type=None*)

Parameters

- **param_id** (*STRING*) – unique identifier for this parameter
- **label** (*STRING*) – short, human-readable label
- **secondary_files** (*STRING*) – If type is a file, describes files that must be included alongside the primary file(s)
- **param_format** (*STRING*) – If type is a file, uri to ontology of the format or exact format.
- **streamable** (*BOOLEAN*) – If type is a file, true indicates that the file is read or written sequentially without seeking
- **doc** (*STRING*) – documentation
- **input_binding** (*cwlgen.CommandLineBinding* object) – describes how to handle the input
- **default** (*STRING*) – default value
- **param_type** (*STRING*) – type of data assigned to the parameter corresponding to CWLType

CommandOutputParameter

class cwlgen.**CommandOutputParameter**(*param_id*, *label=None*, *secondary_files=None*, *param_format=None*, *streamable=None*, *doc=None*, *output_binding=None*, *param_type=None*)

An output parameter for a *cwlgen.CommandLineTool*.

__init__(*param_id*, *label=None*, *secondary_files=None*, *param_format=None*, *streamable=None*, *doc=None*, *output_binding=None*, *param_type=None*)

Parameters

- **param_id** (*STRING*) – unique identifier for this parameter
- **label** (*STRING*) – short, human-readable label
- **secondary_files** (*STRING*) – If type is a file, describes files that must be included alongside the primary file(s)
- **param_format** (*STRING*) – If type is a file, uri to ontology of the format or exact format
- **streamable** (*BOOLEAN*) – If type is a file, true indicates that the file is read or written sequentially without seeking
- **doc** (*STRING*) – documentation
- **output_binding** (*cwlgen.CommandOutputBinding* object) – describes how to handle the output
- **param_type** (*STRING*) – type of data assigned to the parameter corresponding to CWLType

CommandLineBinding

class cwlgen.**CommandLineBinding**(*load_contents=None*, *position=None*, *prefix=None*, *separate=None*, *item_separator=None*, *value_from=None*, *shell_quote=None*)

The binding behavior when building the command line depends on the data type of the value. If there is a mismatch between the type described by the input schema and the effective value, such as resulting from an expression evaluation, an implementation must use the data type of the effective value.

Documentation: https://www.commonwl.org/v1.0/CommandLineTool.html#CommandLineBinding

__init__(*load_contents=None*, *position=None*, *prefix=None*, *separate=None*, *item_separator=None*, *value_from=None*, *shell_quote=None*)

Parameters

- **load_contents** (*BOOLEAN*) – Read up to the fist 64 KiB of text from the file and place it in the "contents" field of the file object
- **position** (*INT*) – The sorting key
- **prefix** (*STRING*) – Command line prefix to add before the value

- **separate** (*BOOLEAN*) –
- **item_separator** (*STRING*) – Join the array elements into a single string separated by this item
- **value_from** (*STRING*) – Use this as the value
- **shell_quote** (*BOOLEAN*) – Value is quoted on the command line

CommandOutputBinding

class cwlgen.**CommandOutputBinding**(*glob=None*, *load_contents=None*, *output_eval=None*)

Describes how to generate an output parameter based on the files produced.

__init__(*glob=None*, *load_contents=None*, *output_eval=None*)

Parameters

- **glob** (*STRING*) – Find corresponding file(s)
- **load_contents** (*BOOLEAN*) – For each file matched, read up to the 1st 64 KiB of text and place it in the contents field
- **output_eval** (*STRING*) – Evaluate an expression to generate the output value

4.2.3 Special Types

class cwlgen.**CommandInputRecordSchema**(*label=None*, *name=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#CommandInputRecordSchema

__init__(*label=None*, *name=None*)

Parameters

- **fields** (*array<InputRecordField>*) – Defines the fields of the record.
- **label** – A short, human-readable label of this object.
- **name** – NF (Name of the InputRecord)

class cwlgen.CommandInputRecordSchema.**CommandInputRecordField**(*name*, *input_type*, *doc=None*, *input_binding=None*, *label=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#CommandInputRecordField

__init__(*name*, *input_type*, *doc=None*, *input_binding=None*, *label=None*)

Parameters

- **name** –
- **input_type** (*CWLType | InputRecordSchema | InputEnumSchema | InputArraySchema | string | array<CWLType | InputRecordSchema | InputEnumSchema | InputArraySchema | string>*) –
- **doc** – A documentation string for this field
- **input_binding** (CommandLineBinding) –
- **label** –

class cwlgen.**CommandInputArraySchema**(*items=None*, *label=None*, *input_binding=None*)

Based on the parameter set out in the CWL spec: https://www.commonwl.org/v1.0/CommandLineTool.html#CommandInputArraySchema

__init__(*items=None*, *label=None*, *input_binding=None*)

Parameters

- **items** – Defines the type of the array elements.
- **label** (*STRING*) – A short, human-readable label of this object.
- **input_binding** (CommandLineBinding) –

Type *CWLType | CommandInputRecordSchema | CommandInputEnumSchema | CommandInputArraySchema | string | array<CWLType | CommandInputRecordSchema | CommandInputEnumSchema | CommandInputArraySchema | string>*

class cwlgen.**CommandInputEnumSchema**(*symbols*, *label=None*, *name=None*, *input_binding=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#CommandInputEnumSchema

__init__(*symbols*, *label=None*, *name=None*, *input_binding=None*)

Parameters

- **symbols** (*List[str]*) – Defines the set of valid symbols.
- **label** (*str*) – A short, human-readable label of this object.
- **name** (*str*) –
- **input_binding** (CommandLineBinding) –

4.3 Workflow API classes

4.3.1 CWL Workflow

Workflow

class cwlgen.workflow.**Workflow**(*workflow_id=None*, *label=None*, *doc=None*, *cwl_version='v1.0'*, *inputs=None*, *outputs=None*, *steps=None*, *requirements=None*, *hints=None*)

A workflow describes a set of steps and the dependencies between those steps. When a step produces output that will be consumed by a second step, the first step is a dependency of the second step.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#Workflow

__init__(*workflow_id=None*, *label=None*, *doc=None*, *cwl_version='v1.0'*, *inputs=None*, *outputs=None*, *steps=None*, *requirements=None*, *hints=None*)

Parameters

- **workflow_id** (*STRING*) – The unique identifier for this process object.
- **label** (*STRING*) – A short, human-readable label of this process object.
- **doc** (*STRING*) – A long, human-readable description of this process object.
- **cwl_version** (*CWLVersion*) – CWL document version. Always required at the document root. Default: 'v1.0'

export(*outfile=None*)

Export the workflow in CWL either on STDOUT or in outfile.

4.3.2 Inputs and Outputs

InputParameter

class cwlgen.workflow.**InputParameter**(*param_id*, *label=None*, *secondary_files=None*, *param_format=None*, *streamable=None*, *doc=None*, *input_binding=None*, *default=None*, *param_type=None*)

Documentation: https://www.commonwl.org/v1.0/Workflow.html#InputParameter

__init__(*param_id*, *label=None*, *secondary_files=None*, *param_format=None*, *streamable=None*, *doc=None*, *input_binding=None*, *default=None*, *param_type=None*)

Parameters

- **param_id** (*STRING*) – unique identifier for this parameter
- **label** (*STRING*) – short, human-readable label
- **secondary_files** (*STRING*) – If type is a file, describes files that must be included alongside the primary file(s)
- **param_format** (*STRING*) – If type is a file, uri to ontology of the format or exact format
- **streamable** (*BOOLEAN*) – If type is a file, true indicates that the file is read or written sequentially without seeking
- **doc** (*STRING*) – documentation
- **param_type** (*STRING corresponding to CWLType*) – type of data assigned to the parameter
- **input_binding** (CommandLineBinding) –

WorkflowStep

class cwlgen.workflow.**WorkflowStep**(*step_id*, *run*, *label=None*, *doc=None*, *scatter=None*, *scatter_method=None*)

A workflow step is an executable element of a workflow. It specifies the underlying process implementation (such as CommandLineTool or another Workflow) in the run field and connects the input and output parameters of the underlying process to workflow parameters.

Documentation: https://www.commonwl.org/v1.0/Workflow.html#WorkflowStep

__init__(*step_id*, *run*, *label=None*, *doc=None*, *scatter=None*, *scatter_method=None*)

Parameters

- **step_id** (*STRING*) – The unique identifier for this workflow step.
- **run** (*STRING | CommandLineTool | ExpressionTool | Workflow*) – Specifies the process to run.
- **label** (*STRING*) – A short, human-readable label of this process object.
- **doc** (*STRING | list[STRING]*) – A long, human-readable description of this process object.
- **scatter** (*STRING | list[STRING]*) – Field to scatter on, see: https://www.commonwl.org/v1.0/Workflow.html#WorkflowStep
- **scatter_method** (*STRING | list[STRING] in [dotproduct, nested_crossproduct, flat_crossproduct]*) – Required if scatter is an array of more than one element.

4.4 Changelogs

Summary of developments of Python-cwlgen library.

4.4.1 v0.3

v0.3.0

This update brings more completeness to the v1.0 spec.

- Large increase in the number of supported classes
- New translation mechanism
- More docstrings

4.4.2 v0.2

v0.2.3

- Bug fix: fix dumping in out_file
- Handle multiline for doc in CWL tool

v0.2.2

- Add namespaces and possibility to add metadata described by schema.org

v0.2.1

- Change order of attribute for CommandLineTool
- remove id field of input and output that appeared in v0.2.0

v0.2.0

- Add import feature for what is covered so far by the library
- Change attribute names of object to correspond exactly to CWL Tool fields

4.4.3 v0.1

v0.1.1

This is the first release of Python-cwlgen:

- Basic model of CWL Tool
- export feature to STDOUT or output file

Index

Symbols

C

D

E

I

M

P

R

S

W