cwltest

Release 2.5.20240304113813.dev6+g7266ad4

Common Workflow Language project and contributors

Apr 25, 2024

CONTENTS

PyPI:

Conda:

This is a testing tool for checking the output of Tools and Workflows described with the Common Workflow Language. Among other uses, it is used to run the CWL conformance tests.

This is written and tested for Python 3.8, 3.9, 3.10, 3.11, and 3.12.

Table of Contents

CHAPTER

ONE

INSTALL

Installing the official package from PyPi

```
pip install cwltest
```

Or from bioconda

```
conda install -c bioconda cwltest
```

Or from source

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

CHAPTER

TWO

RUN ON THE COMMAND LINE

Simple command:

```
cwltest --test test-descriptions.yml --tool cwl-runner
```

CHAPTER

THREE

GENERATE CONFORMANCE BADGES USING CWLTEST

To make badges that show the results of the conformance test, you can generate JSON files for https://badgen.net by using –badgedir option

To generate JSON files:

```
cwltest --test test-descriptions.yml --tool cwl-runner --badgedir badges
...
$ cat badges/command_line_tool.json | jq .
{
  "subject": "command_line_tool",
  "status": "100%",
  "color": "green"
}
```

Once you upload JSON file to a server, you make a badge by using a link like https://badgen.net/https/path/to/generated/json or https://flat.badgen.net/https/path/to/generated/json (for flat badges).

Here is an example of markdown to add a badge:

```
![test result](https://flat.badgen.net/https/path/to/generated/json?icon=commonwl)
```

CHAPTER

FOUR

COMMAND LINE OPTIONS

4.1 cwltest

Common Workflow Language testing framework

```
usage: cwltest [-h] --test TEST [--basedir BASEDIR] [-l] [-n N] [-s S] [-N N]
               [-S S] [--tool TOOL] [--only-tools] [--tags TAGS]
               [--exclude-tags EXCLUDE_TAGS] [--show-tags]
               [--junit-xml JUNIT_XML] [--junit-verbose]
               [--test-arg cache==--cache-dir] [-j J] [--verbose]
               [--classname CLASSNAME] [--timeout TIMEOUT]
               [--badgedir BADGEDIR] [--version]
               ...
```

args
: arguments to pass first to tool runner

-h, --help
: show this help message and exit

--test <test>
: YAML file describing test cases

--basedir <basedir>
: Basedir to use for tests

-l
: List tests then exit

-n <n>
: Run specific tests, format is 1,3-6,9

-s <s>
: Run specific tests using their short names separated by comma

-N <n>
: Exclude specific tests by number, format is 1,3-6,9

-S <s>
: Exclude specific tests by short names separated by comma

--tool <tool>
: CWL runner executable to use (default 'cwl-runner'

--only-tools
: Only test CommandLineTools

--tags <tags>
: Tags to be tested

--exclude-tags <exclude_tags>
: Tags not to be tested

--show-tags
: Show all Tags.

--junit-xml <junit_xml>
: Path to JUnit xml file

--junit-verbose
: Store more verbose output to JUnit XML file by not passing '–quiet' to the CWL runner.

--test-arg <cache==--cache-dir>
: Additional argument given in test cases and required prefix for tool runner.

-j <j>
: Specifies the number of tests to run simultaneously (defaults to one).

--verbose
: More verbose output during test run.

--classname <classname>
: Specify classname for the Test Suite.

--timeout <timeout>
: Time of execution in seconds after which the test will be skipped. Defaults to 600 seconds (10.0 minutes).

--badgedir <badgedir>
: Create JSON badges and store them in this directory.

--version
: show program's version number and exit

4.2 Pytest plugin

`cwltest` can also be used as a Pytest 7.x or 8.x plugin. The CWL test filename must end with `.cwltest.yml` or `.cwltest.yaml`.

In this case, the simple command:

```
cwltest --test conformance_xxx.cwltest.yml --tool cwl-runner
```

becomes:

```
pytest conformance_xxx.cwltest.yml --cwl-runner cwl-runner
```

4.2.1 Command Line Options

pytest

```
usage: pytest [--cwl-runner CWL_RUNNER] [--cwl-runner-verbose]
              [--cwl-badgedir CWL_BADGEDIR] [--cwl-include CWL_INCLUDE]
              [--cwl-exclude CWL_EXCLUDE] [--cwl-tags CWL_TAGS]
              [--cwl-exclude-tags CWL_EXCLUDE_TAGS] [--cwl-args CWL_ARGS]
              [--cwl-test-arg CWL_TEST_ARG] [--cwl-basedir CWL_BASEDIR]
```

--cwl-runner <cwl_runner>
: Name of the CWL runner to use.

--cwl-runner-verbose
: If set, don't pass –quiet to the CWL runner.

--cwl-badgedir <cwl_badgedir>
: Create badge JSON files and store them in this directory.

--cwl-include <cwl_include>
: Run specific CWL tests using their short names separated by comma

--cwl-exclude <cwl_exclude>
: Exclude specific CWL tests using their short names separated by comma

--cwl-tags <cwl_tags>
: Tags to be tested.

--cwl-exclude-tags <cwl_exclude_tags>
: Tags not to be tested.

--cwl-args <cwl_args>
: one or more arguments to pass first to tool runner (separated by spaces)

--cwl-test-arg <cwl_test_arg>
: Additional argument given in test cases and required prefix for tool runner.

--cwl-basedir <cwl_basedir>
: Basedir to use for tests

4.2.2 Converting `cwltest` options to `pytest` options

The table below details all the available command conversions between the two formats.

Feature	cwltest	pytest
YAML file describing test cases	`--test conformance_xxx.cwltest.yml`	`conformance_xxx.cwltest.yml`
CWL runner executable to use	`--tool CWL_RUNNER`	*`--cwl-runner CWL_RUNNER`*
Specifies the number of tests to run simultaneously	`-j CORES`	`-n CORES`[1]
Automatically scale the number of tests to run simultaneously	**UNSUPPORTED**	`-n auto`[Page 12, 1]
Only run one test at a time (good for debugging cwltest itself)	`-j 1` (or leave out `-j`)	`-n 0 -s`[1]
Time of execution in seconds after which the test will be skipped	`--timeout TIMEOUT`	`--timeout TIMEOUT`[3]
List tests then exit	`-l`	`--collect-only`
Run specific tests using their short names	`-s TEST_NAME[,TEST_NAME...]`	*`--cwl-include TEST_NAME[,TEST_NAME...]`*
Exclude specific tests by short names	`-S TEST_NAME[,TEST_NAME...]`	*`--cwl-exclude TEST_NAME[,TEST_NAME...]`*
Tags to be tested	`--tags TAG[,TAG...]`	*`--cwl-tags TAG[,TAG...]`*
Tags not to be tested	`--exclude-tags TAG[,TAG...]`	*`--cwl-exclude-tags TAG[,TAG...]`*
Path to JUnit xml file	`--junit-xml PATH`	`--junit-xml=PATH`[4]
More verbose output during test run	`--verbose`	`-v[vv]`
Additional argument given in test cases and required prefix for tool runner	`--test-arg ARG_NAME==ARG_PREFIX`	*`--cwl-test-arg ARG_NAME==ARG_PREFIX`*
Arguments to pass first to tool runner	`cwltest -- ARG [ARG ...]`	*`--cwl-args "ARG [ARG ...]"`*
Only test CommandLineTools	`--only-tools`	**UNSUPPORTED**
Show all tags	`--show-tags`	**UNSUPPORTED**
Store more verbose output to JUnit xml file	`--junit-verbose`	*`--cwl-runner-verbose`*[4]
Specify classname for the Test Suite	`--classname CLASS_NAME`	**UNSUPPORTED**

4.2.3 Differences in the XML output

`cwltest --junit-xml` output

- top-level <testsuites> element has the elapsed time, and counts (errors, failures, skipped, and total)
- singular <testsuite> sub-element the same attributes as the top-level <testsuites> plus `name` which is the basename of the YAML test file
- each <testcase> element has the follow attributes
  - `name`: the doc string
  - `class`: the tags
  - `file`: the test ID
  - `url`: like "cwltest:conformance_tests#1" (contains the basename of the YAML test file)

[1] Requires pytest-xdist. See Running tests across multiple CPUs.

[3] Requires pytest-timeout. Note: even if `pytest-timeout` is installed, there is no default timeout. This is different than `cwltest`'s default timeout of 10 minutes.

[4] Depending on the value of the pytest INI option `junit_logging`, then <system-out> and <system-err> sub-elements will be generated. However the default value for `junit_logging` is `no`, so to get either of these pick one from the full list. You can set `junit_logging` in a configuration file or on the command line: `pytest -o junit_logging=out-err`.

  - `time`: the elapsed time
- <testcase> elements always contain the following sub-elements, regardless of outcome
  - <system-out>: the output object
  - <system-err>: stderr (docker pull, other warnings, and errors)
- <testcase> elements for failed test cases do not have a <failure> sub-element

`pytest` with `cwltest` plugin XML output

- top-level <testsuites> element has no attributes
- singular <testsuite> sub-element has the same attributes as the `cwltest` XML version along with these additional attributes
  - `name`: default is `pytest` (can be customized with the pytest INI option `junit_suite_name`)
  - `timestamp="2023-01-08T11:39:07.425159"`
  - `hostname`: the hostname of the machine where the tests ran
- inside the <testsuite> is a <properties>..</properties> element with two <property name="..." value="..." /> elements. But this does not work with pytest-xdist.
  - `runner`: the name of the CWL runner
  - `runner_extra_args`: the value of *–cwl-args* `
- each <testcase> element has the following attributes
  - `classname`: always the name of the YAML file (`conformance_test_v1.2.cwltest.yaml`)
  - `name`: the test ID
  - `time`: the elapsed time
- <testcase> elements for failed test cases **do** have a <failure> sub-element with a `message` attribute containing the *cwltest.plugin.CWLItem.repr_failure()* output. This text is repeated as the content of the <failure> element. The presensce of <system-out> and <system-err> sub-elements varies.[4]

4.3 API Reference

This page contains auto-generated API reference documentation[1].

4.3.1 cwltest

Run CWL descriptions with a cwl-runner, and look for expected output.

[1] Created with sphinx-autoapi

Submodules

cwltest.__main__

Default entrypoint for the cwltest module.

cwltest.argparser

Command line argument parsing for cwltest.

Module Contents

Functions

arg_parser()	Generate a command Line argument parser for cwltest.

cwltest.argparser.**arg_parser**()

Generate a command Line argument parser for cwltest.

Return type
argparse.ArgumentParser

cwltest.compare

Compare utilities for CWL objects.

Module Contents

Functions

compare(expected, actual[, skip_details])	Compare two CWL objects.

exception cwltest.compare.**CompareFail**

Bases: Exception

Compared CWL objects are not equal.

classmethod format(*expected*, *actual*, *cause=None*)

Load the difference details into the error message.

Parameters

- **expected** (*Any*)
- **actual** (*Any*)
- **cause** (*Optional[Any]*)

Return type
CompareFail

cwltest.compare.**compare**(*expected*, *actual*, *skip_details=False*)

Compare two CWL objects.

Parameters

- **expected** (*Any*)
- **actual** (*Any*)
- **skip_details** (*bool*)

Return type
None

cwltest.hooks

Hooks for pytest-cwl users.

Module Contents

Functions

pytest_cwl_execute_test(config, processfile, jobfile)	Execute CWL test using a Python function instead of a command line runner.

cwltest.hooks.**pytest_cwl_execute_test**(*config*, *processfile*, *jobfile*)

Execute CWL test using a Python function instead of a command line runner.

The return value is a tuple.

- **status code**
  - 0 = success
  - *cwltest.UNSUPPORTED_FEATURE* for an unsupported feature
  - and any other number for failure
- CWL output object using plain Python objects.

Parameters

- **processfile** (*str*) – a path to a CWL document
- **jobfile** (*Optional[str]*) – an optionl path to JSON/YAML input object

- **config** (*cwltest.utils.CWLTestConfig*)

Return type
Tuple[int, Optional[Dict[str, Any]]]

cwltest.main

Entry point for cwltest.

Module Contents

Functions

main()	Run the main logic loop.

Attributes

PREFIX

cwltest.main.**PREFIX** = '\r'

cwltest.main.**main**()
Run the main logic loop.

Return type
int

cwltest.plugin

Discovers CWL test files and converts them to pytest.Items.

Module Contents

Classes

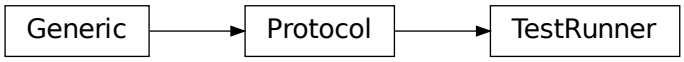
TestRunner	Protocol to type-check test runner functions via the pluggy hook.
CWLItem	A CWL test Item.
CWLYamlFile	A CWL test file.

Functions

pytest_addoption(parser)	Add our options to the pytest command line.
pytest_collect_file(file_path, parent)	Is this file for us.
pytest_configure(config)	Store the raw tests and the test results.
pytest_sessionfinish(session, exitstatus)	Generate badges.
pytest_addhooks(pluginmanager)	Register our cwl hooks.

class cwltest.plugin.**TestRunner**

Bases: Protocol



Protocol to type-check test runner functions via the pluggy hook.

__call__(*config*, *processfile*, *jobfile*)

Type signature for pytest_cwl_execute_test hook results.

Parameters

- **config** (*cwltest.utils.CWLTestConfig*)
- **processfile** (*str*)
- **jobfile** (*Optional[str]*)

Return type

List[Optional[Dict[str, Any]]]

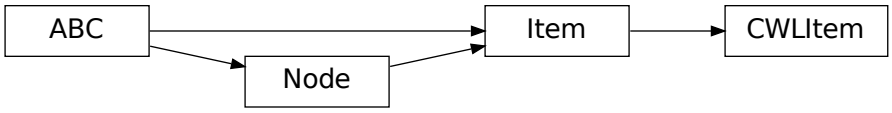
exception cwltest.plugin.**CWLTestException**

Bases: Exception

custom exception for error reporting.

class cwltest.plugin.**CWLItem**(*name*, *parent*, *spec*)

Bases: pytest.Item



A CWL test Item.

Parameters

- **name** (*str*)
- **parent** (*Optional[_pytest.nodes.Node]*)
- **spec** (*Dict[str, Any]*)

`runtest`()

Execute using cwltest.

Return type
None

`repr_failure`(*excinfo*, *style=None*)

Document failure reason.

Called when self.runtest() raises an exception.

Parameters

- **excinfo** (*ExceptionInfo[BaseException]*)
- **style** (*Optional[_pytest._code.code._TracebackStyle]*)

Return type
str

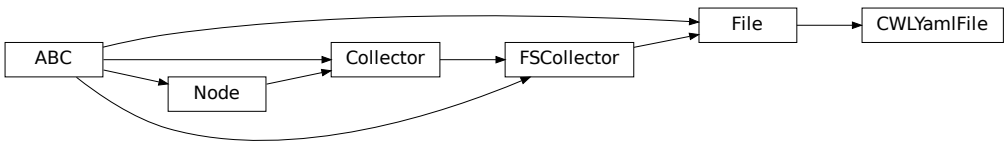
`reportinfo`()

Status report.

Return type
Tuple[Union[os.PathLike[str], str], Optional[int], str]

`class cwltest.plugin.CWLYamlFile`(*fspath=None*, *path_or_parent=None*, *path=None*, *name=None*, *parent=None*, *config=None*, *session=None*, *nodeid=None*)

Bases: `pytest.File`



A CWL test file.

Parameters

- **fspath** (*Optional[_pytest.compat.LEGACY_PATH]*)
- **path_or_parent** (*Optional[Union[pathlib.Path, Node]]*)
- **path** (*Optional[pathlib.Path]*)
- **name** (*Optional[str]*)
- **parent** (*Optional[Node]*)
- **config** (*Optional[_pytest.config.Config]*)
- **session** (*Optional[_pytest.main.Session]*)
- **nodeid** (*Optional[str]*)

collect()

Load the cwltest file and yield parsed entries.

Return type
Iterator[*CWLItem*]

cwltest.plugin.**pytest_addoption**(*parser*)

Add our options to the pytest command line.

Parameters
parser (*_pytest.config.argparsing.Parser*)

Return type
None

cwltest.plugin.**pytest_collect_file**(*file_path*, *parent*)

Is this file for us.

Parameters

- **file_path** (*pathlib.Path*)
- **parent** (*pytest.Collector*)

Return type
Optional[pytest.Collector]

cwltest.plugin.**pytest_configure**(*config*)

Store the raw tests and the test results.

Parameters
config (*_pytest.config.Config*)

Return type
None

cwltest.plugin.**pytest_sessionfinish**(*session*, *exitstatus*)

Generate badges.

Parameters

- **session** (*pytest.Session*)
- **exitstatus** (*int*)

Return type
None

cwltest.plugin.**pytest_addhooks**(*pluginmanager*)

Register our cwl hooks.

Parameters
pluginmanager (*_pytest.config.PytestPluginManager*)

Return type
None

cwltest.utils

Package Contents

cwltest.**UNSUPPORTED_FEATURE = 33**

cwltest.**DEFAULT_TIMEOUT = 600**

cwltest.**REQUIRED = 'required'**

cwltest.**logger**

cwltest.**templock**

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

Symbols

A

C

D

F

L

M

P

R

T

U