
SevenBridges CWL Documentation

Release 0.0.1

Filip Tubic

Nov 23, 2018

Contents:

1	Installation	1
2	Get the Code	3
3	Quickstart	5
3.1	Overview	5
3.2	Tool creation	5
3.3	Workflow creation	8
3.4	Loading existing documents	10
4	SevenBridges CWL package	11
4.1	Submodules	34
4.2	sbg.cwl.sbg.session module	34
4.3	sbg.cwl.v1_0.schema module	36
4.4	sbg.cwl.v1_0.cmd module	40
4.5	sbg.cwl.v1_0.wf module	44
5	Indices and tables	51
	Python Module Index	53

CHAPTER 1

Installation

The easiest way to install sevenbridges-cwl is using pip:

```
$ pip install sevenbridges-cwl
```


CHAPTER 2

Get the Code

Library `sevenbridges-cwl` is actively developed on GitHub.

The easiest way to obtain the source is to clone the public repository:

```
$ git clone git@github.com:sbg/sevenbridges-cwl.git
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages by invoking:

```
$ python setup.py install
```


3.1 Overview

SevenBridges CWL package provides python bindings for *Common Workflow Language v1.0*. It is intended for developers who want to use python code to generate CWL documents. If creating a document through the GUI is preferable, then look at the [Rabix Composer](#). This library also have integration with [sevenbridges-python](#) so applications can be easily deployed on a Seven Bridges platform.

3.2 Tool creation

3.2.1 Docker image preparation

Your code will run in a docker image that you have prepared before defining and running a tool. All libraries needed for your function to run must be installed on the image. The only additional requirement for `sevenbridges-cwl` is to have `dill` package and `bzip2` installed on the image - so when you prepare your image `pip install dill` and `apt-get install bzip2` on it.

3.2.2 Wrapping binaries

One way of creating a tool is by using `CommandLineTool` class which is useful for wrapping binaries:

```
from sbg import cwl

t = cwl.CommandLineTool(
    base_command=['echo', 'HelloWorld'],
    stdout='_stdout_',
    requirements=[cwl.Docker(docker_pull='ubuntu:16.04')]
)
t.add_output(cwl.File(glob='_stdout_', required=True), id='out')
```

Example above illustrates `echo HelloWorld > _stdout_` command. Object `t` is an instance of `CommandLineTool` class which has a number of useful builtin methods described [here](#). Generated tool can be easily run on a Seven Bridges platform using `Session` object:

```
session = cwl.Session(profile='<your_profile>')
session.run('<your_project>', t)
```

If inspecting raw CWL documents is preferable use `cwl.tool` context manager:

```
from sbg import cwl

with cwl.tool('hello_world.cwl', 'w') as t:
    t.base_command = ['echo', 'HelloWorld'] # echo 'HelloWorld' on stdout
    t.add_requirement(cwl.Docker(docker_pull='ubuntu:16.04'))
    t.stdout = '_stdout_' # redirect all stdout to this '_stdout_' file
    t.add_output(cwl.File(glob='_stdout_', required=True), id='out')
```

First parameter to the `tool` function is a file path for CWL document. Second parameter is file access which can be either one of:

- `w` - for writing
- `r` - for reading
- `rw` - for editing

After running code block above, `hello_world.cwl` file is created and dumped into the current working directory with contents:

```
baseCommand:
- echo
- HelloWorld
class: CommandLineTool
cwlVersion: v1.0
inputs: []
outputs:
- id: out
  outputBinding:
    glob: _stdout_
    type: File
requirements:
- class: DockerRequirement
  dockerPull: ubuntu:16.04
stdout: _stdout_
```

3.2.3 Wrapping python code

Tool can be created using `@to_tool` decorator only by *annotating* python function. Annotated functions are functions with defined types for inputs and outputs, which is illustrated in the example below.

```
import pysam
from sbg import cwl

@cwl.to_tool(
    docker='images.sbggenomics.com/filip_tubic/ubuntu1604pysam',
```

(continues on next page)

(continued from previous page)

```

    outputs=dict(out=cwl.File(glob='gc_content.txt'))
)
def gc_content(bam_file: cwl.File(secondary_files=['.bai']),
               bed_file: cwl.File()):
    """Calculates GC content."""

    bam_file = bam_file['path']
    bed_file = bed_file['path']
    bam = pysam.AlignmentFile(bam_file, 'rb')
    with open('gc_content.txt', 'w') as out:
        with open(bed_file) as bf:
            for line in bf:
                line_parts = line.strip().split()
                chr = line_parts[0]
                start = int(line_parts[1])
                end = int(line_parts[2])
                read_data = bam.fetch(chr, start, end)
                total_bases = 0
                gc_bases = 0
                for read in read_data:
                    seq = read.query_sequence
                    total_bases += len(seq)
                    gc_bases += len([x for x in seq if x == 'C' or x == 'G'])
                if total_bases == 0:
                    gc_percent = 'No Reads'
                else:
                    gc_percent = '{0:.2f}%'.format(
                        float(gc_bases) / total_bases * 100
                    )
                out.write('{0}\t{1}\n'.format(line.strip(), gc_percent))

```

Function `gc_content` will accept `.bam` and `.bed` files and calculates GC content for each interval defined in bed file. Corresponding output will be dumped into a `gc_content.txt` file. After running code above, command line tool will be created with already set inputs (`bam_file`, `bed_file`) and output (`out`). In order to run this function we use `Session`.

```

session = cwl.Session(profile='<your_profile>')
project = '<your_project>'

files = list(session.api.files.query(
    project=project,
    names=['<bam_file>', '<bed_file>']
))

session.run(project, gc_content(), inputs=dict(
    bam_file=files[0],
    bed_file=files[1]
))

```

NOTE After generating tool from `gc_content` function, base command will be set to `python{major}.{minor} gc_content.py` where `{major}.{minor}` is python version that is used for calling code block above. So if you're using python 3.6 locally you need to have python 3.6 installed in your docker image.

Input/Output types are translated into CWL concrete types by following rules:

- `cwl.Int()` is converted into cwl integer

- `cwl.String()` is converted into cwl string
- `cwl.Float()` is converted into cwl float
- `cwl.Bool()` is converted into cwl boolean
- `cwl.File()` is converted into cwl file
- `cwl.Dir()` is converted into cwl directory
- `cwl.Record(k1=cwl.String(), k2=cwl.Int())` is converted into cwl record with string and int as field types named k1 and k2 respectively
- `cwl.Union()` is converted into Union type (can be either one of specified types, eg: `cwl.Union([cwl.Int(), cwl.String()])` - int or string)
- `cwl.Enum()` is converted into cwl enum
- `cwl.Array(<t>)` is converted into cwl array of type t (eg. `cwl.Array(cwl.Int())` - list of ints)

Complete documentation of `@to_tool` decorator is located [here](#).

3.2.4 Wrapping bash code

Tools can be generated from existing bash scripts using `cwl.from_bash` function.

```
from sbg import cwl

t = cwl.from_bash(
    label='Example of bash tool',
    inputs=dict(
        STR=cwl.String(),
    ),
    outputs=dict(
        out=cwl.File(glob='stdout')
    ),
    script=r'''echo $STR''',
    stdout='stdout',
    docker='images.sbggenomics.com/filip_tubic/ubuntu1604bzip'
)
```

3.3 Workflow creation

Workflow can be easily created from existing tool objects. One way of creating workflow can be done using with `workflow(...)` statement.

```
from sbg import cwl

# First node
@cwl.to_tool(
    inputs=dict(x=cwl.String()),
    outputs=dict(out=cwl.Float(required=True)),
    docker='images.sbggenomics.com/filip_tubic/ubuntu1604py'
)
def to_float(x):
```

(continues on next page)

(continued from previous page)

```

    return dict(out=float(x))

# Second node
@cwl.tool(
    inputs=dict(x=cwl.Float(), n=cwl.Int()),
    outputs=dict(out=cwl.Float()),
    docker='images.sbgenomics.com/filip_tubic/ubuntu1604py'
)
def times_n(x, n=10):
    return dict(out=x * n)

with cwl.workflow('wf.cwl', 'w') as wf:
    # create tools
    t1 = to_float()
    t2 = times_n()

    # steps
    wf.add_step(t1, expose=['x'])
    wf.add_step(t2, expose=['n', 'out'])

    # add connections
    wf.add_connection('{} .out'.format(t1.id), '{}.x'.format(t2.id))

```

Object wf is an instance of Workflow class which documentation can be found [here](#).

Running code block above will generate wf.cwl in the current working directory. Using [Rabix Composer](#) generated file can be easily visualized as a graph. By pasting contents of wf.cwl in the Code section in Rabix composer, following [graph](#) will be displayed in Visual Editor section. Like in examples before, we use Session to run workflow.

```

from sbg import cwl

# First node
@cwl.tool(
    inputs=dict(x=cwl.String()),
    outputs=dict(out=cwl.Float(required=True)),
    docker='images.sbgenomics.com/filip_tubic/ubuntu1604py'
)
def to_float(x):
    return dict(out=float(x))

# Second node
@cwl.tool(
    inputs=dict(x=cwl.Float(), n=cwl.Int()),
    outputs=dict(out=cwl.Float()),
    docker='images.sbgenomics.com/filip_tubic/ubuntu1604py'
)
def times_n(x, n=10):
    return dict(out=x * n)

wf = cwl.Workflow()

```

(continues on next page)

(continued from previous page)

```
# create tools
t1 = to_float()
t2 = times_n()

# steps
wf.add_step(t1, expose=['x'])
wf.add_step(t2, expose=['n', 'out'])

# add connections
wf.add_connection('{}out'.format(t1.id), '{}.x'.format(t2.id))

# Session on a SBG platform
session = cwl.Session(profile='<your_profile>')

session.run('<your_project>', wf, inputs={'x': '10.2', 'n': 10})
```

3.4 Loading existing documents

Existing CWL documents can be loaded from a file using `load` function, [docs](#).

```
from sbg import cwl

t = cwl.CommandLineTool(
    base_command=['echo', 'Hello'],
).dump('dummy.cwl')

x = cwl.load('dummy.cwl')
print(' '.join(x.base_command)) # prints 'echo Hello'

assert isinstance(x, cwl.CommandLineTool)
```

SevenBridges CWL package

`sbg.cwl.v1_0.inherit_metadata(self, src, dst)`
 Inherits metadata from `src` input id to `dst` output id.

Parameters

- **src** – input ID
- **dst** – output ID

class `sbg.cwl.v1_0.Cwl`

Bases: `dict`

Super class for all CWL v1.0 subclasses.

to_json()

Returns serialized JSON representation for this object.

Returns serialized object

to_dict()

calc_hash()

Returns calculated hash value for this object.

Returns hash value using `hashlib.sha512` encoded with `utf-8`

dump(path)

Dump this object into file formatted as YAML.

Parameters `path` – file path

json_dump(path)

Dump this object into file formatted as JSON.

Parameters `path` – file path

resolve()

Resolve all salad schema \$directives inside object (\$mixin, \$include, \$import).

`sbg.cwl.v1_0.load(cwl)`

Loads CWL document from file or JSON object and instantiate object of a class specified by key `class` inside a document.

Parameters `cwl` – file (can be either in JSON or YAML format)

Returns depending on `class` can be either an instance of `CommandLineTool` or `ExpressionTool` or `Workflow`

class `sbg.cwl.v1_0.Primitive`

Bases: `object`

NULL = `'null'`

BOOLEAN = `'boolean'`

INT = `'int'`

LONG = `'long'`

FLOAT = `'float'`

DOUBLE = `'double'`

STRING = `'string'`

FILE = `'File'`

DIRECTORY = `'Directory'`

ANY = `'Any'`

`sbg.cwl.v1_0.is_number(t)`

Checks by CWL v1.0 specification if object `t` is number type.

`sbg.cwl.v1_0.is_primitive(t)`

Checks by CWL v1.0 specification if object `t` is primitive type.

`sbg.cwl.v1_0.tool_from(f, path=None, access='r')`

Class that can be used with `with` statement for reading/writing/editing tool from function `f`.

Parameters

- **f** – function
- **path** – file path
- **access** – access permission (`'r'` - read, `'w'` - write, `'rw'` - edit)

Example

```
from sbg import cwl

def multiply(x: cwl.Int(), y: cwl.Int()) -> dict(out=cwl.Int()):
    return dict(out=x * y)

# creates CWL application from function multiply
with cwl.tool_from(multiply, 'multiply.cwl', 'w') as t:
    t.add_docker_requirement(
        docker_pull='<docker_with_python>'
    )
```

`sbj.cwl.v1_0.tool(path=None, access='r')`

Class that can be used with `with` statement for reading/writing/editing tool.

Parameters

- **path** – file path
- **access** – access permission ('r' - read, 'w' - write, 'rw' - edit)

Example of writing:

```
from sbj import cwl

with cwl.tool('example.cwl', 'w') as t:
    t.id = 'tool_id' # set id
    t.label = 'Dummy tool' # set label
```

Example of reading:

```
from sbj import cwl

with cwl.tool('example.cwl', 'r') as t:
    print(' '.join(t.base_command)) # print base command
```

Example of editing:

```
from sbj import cwl

with cwl.tool('example.cwl', 'rw') as t:
    t.doc = 'New description...' # edit tool description
```

`sbj.cwl.v1_0.workflow(path=None, access='r')`

Class that can be used with `with` statement for reading/writing/editing workflow.

Parameters

- **path** – file path
- **access** – access permission ('r' - read, 'w' - write, 'rw' - edit)

Example of writing:

```
from sbj import cwl

with cwl.workflow('example.cwl', 'w') as wf:
    wf.id = 'workflow_id' # set id
    wf.label = 'Dummy workflow' # set label
```

Example of reading:

```
from sbj import cwl

with cwl.workflow('example.cwl', 'r') as wf:
    for step in wf.steps: # print workflow steps
        print(step)
```

Example of editing:

```
from sbj import cwl
```

(continues on next page)

(continued from previous page)

```
with cwl.workflow('example.cwl', 'rw') as wf:
    wf.doc = 'New description...' # edit workflow description
```

```
sbgl.cwl.v1_0.to_tool(inputs=None, outputs=None, id=None, requirements=None, hints=None,
                      label=None, doc=None, base_command=None, arguments=None,
                      stdin=None, stdout=None, stderr=None, success_codes=None, temporary_fail_codes=None, permanent_fail_codes=None, docker=None, js=True,
                      sh=True)
```

```
@sbgl.cwl.v1_0.to_tool
```

Decorator for creating tool from a function.

Parameters

- **inputs** – annotated inputs represented as a dictionary, where keys=inputs and values=types
- **outputs** – annotated outputs represented as a dictionary, where keys=outputs and values=types
- **id** – id of a tool (default name of a decorated function)
- **requirements** – list of a tool requirements
- **hints** – list of a tool hints
- **label** – label of a tool
- **doc** – description of a tool
- **base_command** – base command of a tool
- **arguments** – command line bindings which are not directly associated with input parameters
- **stdin** – a path to a file whose contents must be piped into the command's standard input stream
- **stdout** – capture the command's standard output stream to a file written to the designated output directory
- **stderr** – capture the command's standard error stream to a file written to the designated output directory.
- **success_codes** – exit codes that indicate the process completed successfully.
- **temporary_fail_codes** – Exit codes that indicate the process failed due to a possibly temporary condition
- **permanent_fail_codes** – exit codes that indicate the process failed due to a permanent logic error
- **docker** – specify a Docker image to retrieve using docker pull
- **js** – include `InlineJavascriptRequirement`
- **sh** – include `ShellCommandRequirement`

Returns typing.Callable[... , CommandLineTool]

Example:

```

from sbg import cwl

@cwl.to_tool(
    inputs=dict(x=cwl.Float(), n=cwl.Int()),
    outputs=dict(out=cwl.Float()),
    docker='images.sbggenomics.com/filip_tubic/ubuntu1604py'
)
def times_n(x, n=10):
    """Returns x * n"""
    return dict(out=x * n)

t = times_n()

print(t.label) # prints 'to_tool'
print(t.doc) # prints 'Returns x * n'
print(t.inputs) # prints inputs
print(t.outputs) # prints outputs
print(t.base_command) # prints [python{major}.{minor}, 'to_tool.py']

```

```

class sbg.cwl.v1_0.CommandInput(id=None, label=None, secondary_files=None, stream-
                                able=None, doc=None, format=None, input_binding=None,
                                default=None, type=None)

```

Bases: `sbg.cwl.v1_0.base.Cwl`

An input parameter for a CommandLineTool.

id

The unique identifier for this parameter object.

label

A short, human-readable label of this object.

secondary_files

Only valid when type – File or is an array of items File.

Provides a pattern or expression specifying files or directories that must be included alongside the primary file. All listed secondary files must be present. An implementation may fail wf execution if an expected secondary file does not exist.

If the value is an expression, the value of self in the expression must be the primary input or output File object to which this binding applies. The basename, nameroot and nameext fields must be present in self. For CommandLineTool outputs the path field must also be present. The expression must return a filename string relative to the path to the primary File, a File or Directory object with either path or location and basename fields set, or an array consisting of strings or File or Directory objects. It is legal to reference an unchanged File or Directory object taken from input as a secondaryFile.

To work on non-filename-preserving storage systems, portable tool descriptions should avoid constructing new values from location, but should construct relative references using basename or nameroot instead.

If a value in secondaryFiles is a string that is not an expression, it specifies that the following pattern should be applied to the path of the primary file to yield a filename relative to the primary File:

- If string begins with one or more caret ^ characters, for each caret, remove the last file extension from the path (the last period . and all following characters). If there are no file extensions, the path is unchanged.
- Append the remainder of the string to the end of the file path.

streamable

Only valid when type File or is an array of items File.

A value of true indicates that the file is read or written sequentially without seeking. An implementation may use this flag to indicate whether it is valid to stream file contents using a named pipe. Default: false.

doc

A documentation string for this type, or an array of strings which should be concatenated.

format

Only valid when type File or is an array of items File.

This must be one or more IRIs of concept nodes that represents file formats which are allowed as input to this parameter, preferably defined within an ontology. If no ontology is available, file formats may be tested by exact match.

input_binding

Describes how to handle the inputs of a process and convert them into a concrete form for execution, such as command line parameters.

default

The default value to use for this parameter if the parameter is missing from the input object, or if the value of the parameter in the input object is null. Default values are applied before evaluating expressions (e.g. dependent valueFrom fields).

type

Specify valid types of data that may be assigned to this parameter.

```
class sbg.cwl.v1_0.CommandLineTool (cwl_version='v1.0', inputs=None, outputs=None,
                                     id=None, requirements=None, hints=None, la-
                                     bel=None, doc=None, base_command=None, argu-
                                     ments=None, stdin=None, stdout=None, stderr=None,
                                     success_codes=None, temporary_fail_codes=None,
                                     permanent_fail_codes=None, **kwargs)
```

Bases: sbg.cwl.v1_0.app.App

This defines the schema of the CWL Command Line Tool Description document.

```
class_ = 'CommandLineTool'
```

```
unarchive_bundle (bundle, encoded=False, postprocess=None)
```

Adds first unarchiving command as first argument. It can also decode base64 encoded bundles.

Parameters

- **bundle** – bundle name
- **encoded** – flag which indicates that bundle is base64 encoded

```
static get_unarchive_argument (bundle, encoded=False, postprocess=None)
```

Returns InputBinding argument with untar command.

Parameters

- **bundle** – archive name
- **encoded** – flag which indicates that bundle is base64 encoded

Returns an instance of InputBinding which is an argument

```
add_locals (locals, name, postprocess=None)
```

Add local files/dirs to tool in runtime.

Parameters

- **locals** – list with paths
- **name** – bundle name in runtime

- **postprocess** – bash operation after unarchiving a bundle

classmethod from_bash (*script*, *name*='script.sh', *id*=None, *label*=None, *doc*=None, *inputs*=None, *outputs*=None, *sources*=None, *lib*=None, *docker*=None, *secondary_files*=None, *stdout*=None)

Creates CommandLineTool created from bash script.

Parameters

- **script** – can be bash file or contents
- **name** – bash script name
- **id** – tool ID
- **label** – tool label
- **doc** – tool description
- **inputs** – dictionary where keys are variable names and values can be either one of cwl hints or string. if hint is specified, then corresponding input will be created. If string is specified, then environment variable with that string value will be created.
- **outputs** – dictionary where keys are output ids and values are cwl hints
- **sources** – executes `source <source>` which can be used for exporting bash variables
- **lib** – only required when sources are specified, used as a name for bundle with sources
- **docker** – specify a docker image to retrieve using docker pull
- **secondary_files** – dictionary where key is input/output id and value is secondary files
- **stdout** – standard output redirect to this file

Returns an instance of `cwl.CommandLineTool`

get_requirements (*value*)

base_command

Specifies the program to execute. If an array, the first element of the array is the command to execute, and subsequent elements are mandatory command line arguments. The elements in `baseCommand` must appear before any command line bindings from `inputBinding` or arguments.

If `baseCommand` is not provided or is an empty array, the first element of the command line produced after processing `inputBinding` or arguments must be used as the program to execute.

If the program includes a path separator character it must be an absolute path, otherwise it is an error. If the program does not include a path separator, search the `$PATH` variable in the runtime environment of the wf runner find the absolute path of the executable.

arguments

Command line bindings which are not directly associated with input parameters.

stdin

A path to a file whose contents must be piped into the command's standard input stream.

stdout

Capture the command's standard output stream to a file written to the designated output directory.

If `stdout` is a string, it specifies the file name to use.

If stdout is an expression, the expression is evaluated and must return a string with the file name to use to capture stdout. If the return value is not a string, or the resulting path contains illegal characters (such as the path separator /) it is an error.

stderr

Capture the command's standard error stream to a file written to the designated output directory.

If stderr is a string, it specifies the file name to use.

If stderr is an expression, the expression is evaluated and must return a string with the file name to use to capture stderr. If the return value is not a string, or the resulting path contains illegal characters (such as the path separator /) it is an error.

success_codes

Exit codes that indicate the process completed successfully.

temporary_fail_codes

Exit codes that indicate the process failed due to a possibly temporary condition, where executing the process with the same runtime environment and inputs may produce different results.

permanent_fail_codes

Exit codes that indicate the process failed due to a permanent logic error, where executing the process with the same runtime environment and same inputs is expected to always fail.

```
class sbg.cwl.v1_0.CommandOutput (id=None, label=None, secondary_files=None, stream-  
able=None, doc=None, output_binding=None, for-  
mat=None, type=None)
```

Bases: `sbg.cwl.v1_0.base.Cwl`

An output parameter for a CommandLineTool.

id

The unique identifier for this parameter object.

label

A short, human-readable label of this object.

secondary_files

Only valid when type File or is an array of items File.

Provides a pattern or expression specifying files or directories that must be included alongside the primary file. All listed secondary files must be present. An implementation may fail wf execution if an expected secondary file does not exist.

If the value is an expression, the value of self in the expression must be the primary input or output File object to which this binding applies. The basename, nameroot and nameext fields must be present in self. For CommandLineTool outputs the path field must also be present. The expression must return a filename string relative to the path to the primary File, a File or Directory object with either path or location and basename fields set, or an array consisting of strings or File or Directory objects. It is legal to reference an unchanged File or Directory object taken from input as a secondaryFile.

To work on non-filename-preserving storage systems, portable tool descriptions should avoid constructing new values from location, but should construct relative references using basename or nameroot instead.

If a value in secondaryFiles is a string that is not an expression, it specifies that the following pattern should be applied to the path of the primary file to yield a filename relative to the primary File

1. If string begins with one or more caret ^ characters, for each caret, remove the last file extension from the path (the last period . and all following characters). If there are no file extensions, the path is unchanged.
2. Append the remainder of the string to the end of the file path.

streamable

Only valid when type File or is an array of items File.

A value of true indicates that the file is read or written sequentially without seeking. An implementation may use this flag to indicate whether it is valid to stream file contents using a named pipe. Default false.

doc

A documentation string for this type, or an array of strings which should be concatenated.

output_binding

Describes how to handle the outputs of a process.

format

Only valid when type File or is an array of items File.

This is the file format that will be assigned to the output parameter.

type

Specify valid types of data that may be assigned to this parameter.

```
class sbg.cwl.v1_0.EnvVar (env_def)
```

Bases: sbg.cwl.v1_0.base.Cwl

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

```
class_ = 'EnvVarRequirement'
```

env_def

The list of environment variables.

```
class sbg.cwl.v1_0.EnvironmentDef (env_name, env_value)
```

Bases: sbg.cwl.v1_0.base.Cwl

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

env_name

The environment variable name

env_value

The environment variable name

```
class sbg.cwl.v1_0.SchemaDef (types)
```

Bases: sbg.cwl.v1_0.base.Cwl

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.

```
class_ = 'SchemaDefRequirement'
```

types

The list of type definitions.

```
class sbg.cwl.v1_0.Software (packages)
```

Bases: sbg.cwl.v1_0.base.Cwl

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

```
class_ = 'SchemaDefRequirement'
```

packages

The (optional) versions of the software that are known to be compatible.

```
class sbg.cwl.v1_0.SoftwarePackage (package, version=None, specs=None)
```

Bases: sbg.cwl.v1_0.base.Cwl

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 named in the package field. Implementations may provide resolvers which map these software identifier IRIs to some configuration action; or they can use only the name from the package field on a best effort basis.

For example, the IRI <https://packages.debian.org/bowtie> could be resolved with `apt-get install bowtie`. The IRI <https://anaconda.org/bioconda/bowtie> could be resolved with `conda install -c bioconda bowtie`.

IRIs can also be system independent and used to map to a specific software installation or selection mechanism. Using RRID as an example: https://identifiers.org/rrid/RRID:SCR_005476 could be fulfilled using the above mentioned Debian or bioconda package, a local installation managed by Environment Modules, or any other mechanism the platform chooses. IRIs can also be from identifier sources that are discipline specific yet still system independent. As an example, the equivalent ELIXIR Tools and Data Service Registry IRI to the previous RRID example is <https://bio.tools/tool/bowtie2/version/2.2.8>. If supported by a given registry, implementations are encouraged to query these system independent software identifier IRIs directly for links to packaging systems.

A site specific IRI can be listed as well. For example, an academic computing cluster using Environment Modules could list the IRI <https://hpc.example.edu/modules/bowtie-tbb/1.22> to indicate that module load `bowtie-tbb/1.1.2` should be executed to make available bowtie version 1.1.2 compiled with the TBB library prior to running the accompanying Workflow or CommandLineTool. Note that the example IRI is specific to a particular institution and computing environment as the Environment Modules system does not have a common namespace or standardized naming convention.

This last example is the least portable and should only be used if mechanisms based off of the package field or more generic IRIs are unavailable or unsuitable. While harmless to other sites, site specific software IRIs should be left out of shared CWL descriptions to avoid clutter.

```
class sbg.cwl.v1_0.InitialWorkDir (listing)
```

```
    Bases: sbg.cwl.v1_0.base.Cwl
```

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

```
class_ = 'InitialWorkDirRequirement'
```

listing

The list of files or subdirectories that must be placed in the designated output directory prior to executing the command line tool.

May be an expression. If so, the expression return value must validate as {type: array, items: [File, Directory]}.

Files or Directories which are listed in the input parameters and appear in the InitialWorkDirRequirement listing must have their path set to their staged location in the designated output directory. If the same File or Directory appears more than once in the InitialWorkDirRequirement listing, the implementation must choose exactly one value for path; how this value is chosen is undefined.

```
class sbg.cwl.v1_0.Dirent (entry, entryname=None, writable=None)
```

```
    Bases: sbg.cwl.v1_0.base.Cwl
```

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.

entry

If the value is a string literal or an expression which evaluates to a string, a new file must be created with the string as the file contents.

If the value is an expression that evaluates to a File object, this indicates the referenced file should be added to the designated output directory prior to executing the tool.

If the value is an expression that evaluates to a Dired object, this indicates that the File or Directory in entry should be added to the designated output directory with the name in entryname.

If writable is false, the file may be made available using a bind mount or file system link to avoid unnecessary copying of the input file.

entryname

The name of the file or subdirectory to create in the output directory. If entry is a File or Directory, the entryname field overrides the value of basename of the File or Directory object. Optional.

writable

If true, the file or directory must be writable by the tool. Changes to the file or directory must be isolated and not visible by any other CommandLineTool process. This may be implemented by making a copy of the original file or directory. Default false (files and directories read-only by default).

A directory marked as writable: true implies that all files and subdirectories are recursively writable as well.

```
class sbg.cwl.v1_0.Docker (docker_pull=None,      docker_load=None,      docker_file=None,
                          docker_import=None,      docker_image_id=None,
                          docker_output_directory=None)
Bases: sbg.cwl.v1_0.base.Cwl
```

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

If a CommandLineTool lists DockerRequirement under hints (or requirements), it may (or must) be run in the specified Docker container.

The platform must first acquire or install the correct Docker image as specified by dockerPull, dockerImport, dockerLoad or dockerFile.

The platform must execute the tool in the container using docker run with the appropriate Docker image and tool command line.

The wf platform may provide input files and the designated output directory through the use of volume bind mounts. The platform may rewrite file paths in the input object to correspond to the Docker bind mounted locations.

When running a tool contained in Docker, the wf platform must not assume anything about the contents of the Docker container, such as the presence or absence of specific software, except to assume that the generated command line represents a valid command within the runtime environment of the container.

Interaction with other requirements

If EnvVarRequirement is specified alongside a DockerRequirement, the environment variables must be provided to Docker using --env or --env-file and interact with the container's preexisting environment as defined by Docker.

```
class_ = 'DockerRequirement'
```

docker_pull

Specify a Docker image to retrieve using docker pull.

docker_load

Specify a HTTP URL from which to download a Docker image using docker load.

docker_file

Supply the contents of a Dockerfile which will be built using docker build.

docker_import

Provide HTTP URL to download and gunzip a Docker images using docker import.

docker_image_id

The image id that will be used for docker run. May be a human-readable image name or the image identifier hash. May be skipped if dockerPull is specified, in which case the dockerPull image id must be used.

docker_output_directory

Set the designated output directory to a specific location inside the Docker container.

```
class sbg.cwl.v1_0.InlineJavascript (expression_lib=None)
```

Bases: sbg.cwl.v1_0.base.Cwl

Indicates that the wf platform must support inline Javascript expressions. If this requirement is not present, the wf platform must not perform expression interpolation.

```
class_ = 'InlineJavascriptRequirement'
```

expression_lib

Additional code fragments that will also be inserted before executing the expression code. Allows for function definitions that may be called from CWL expressions.

```
class sbg.cwl.v1_0.Resource (cores_min=None, cores_max=None, ram_min=None,  
ram_max=None, tmpdir_min=None, tmpdir_max=None, out-  
dir_min=None, outdir_max=None)
```

Bases: sbg.cwl.v1_0.base.Cwl

Specify basic hardware resource requirements.

“min” is the minimum amount of a resource that must be reserved to schedule a job. If “min” cannot be satisfied, the job should not be run.

“max” is the maximum amount of a resource that the job shall be permitted to use. If a node has sufficient resources, multiple jobs may be scheduled on a single node provided each job’s “max” resource requirements are met. If a job attempts to exceed its “max” resource allocation, an implementation may deny additional resources, which may result in job failure.

If “min” is specified but “max” is not, then “max” == “min” If “max” is specified by “min” is not, then “min” == “max”.

It is an error if max < min.

It is an error if the value of any of these fields is negative.

If neither “min” nor “max” is specified for a resource, an implementation may provide a default.

```
class_ = 'ResourceRequirement'
```

cores_min

Minimum reserved number of CPU cores

cores_max

Maximum reserved number of CPU cores

ram_min

Minimum reserved RAM in mebibytes (2**20)

ram_max

Maximum reserved RAM in mebibytes (2**20)

tmpdir_min

Minimum reserved filesystem based storage for the designated temporary directory, in mebibytes (2**20)

tmpdir_max

Maximum reserved filesystem based storage for the designated temporary directory, in mebibytes (2**20)

outdir_min

Minimum reserved filesystem based storage for the designated output directory, in mebibytes (2**20)

outdir_max

Maximum reserved filesystem based storage for the designated output directory, in mebibytes (2**20)

class sbg.cwl.v1_0.ShellCommand

Bases: sbg.cwl.v1_0.base.Cwl

Modify the behavior of CommandLineTool to generate a single string containing a shell command line. Each item in the argument list must be joined into a string separated by single spaces and quoted to prevent interpretation by the shell, unless CommandLineBinding for that argument contains shellQuote: false. If shellQuote: false is specified, the argument is joined into the command string without quoting, which allows the use of shell metacharacters such as | for pipes.

class_ = 'ShellCommandRequirement'

class sbg.cwl.v1_0.WorkflowInput (*id=None, label=None, secondary_files=None, streamable=None, doc=None, format=None, input_binding=None, default=None, type=None*)

Bases: sbg.cwl.v1_0.cmd.input.CommandInput

class sbg.cwl.v1_0.MergeMethod

Bases: object

MERGE_NESTED = 'merge_nested'

MERGE_FLATTENED = 'merge_flattened'

class sbg.cwl.v1_0.WorkflowOutput (*id=None, label=None, secondary_files=None, streamable=None, doc=None, output_binding=None, format=None, type=None, output_source=None, link_merge=None*)

Bases: sbg.cwl.v1_0.cmd.output.CommandOutput

output_source

Specifies one or more wf parameters that supply the value of to the output parameter.

link_merge

The method to use to merge multiple sources into a single array. If not specified, the default method is "merge_nested".

class sbg.cwl.v1_0.StepInput (*id, source=None, link_merge=None, default=None, value_from=None*)

Bases: sbg.cwl.v1_0.base.Cwl

The input of a wf step connects an upstream parameter (from the wf inputs, or the outputs of other workflows steps) with the input parameters of the underlying step.

Input object

A WorkflowStepInput object must contain an id field in the form #fieldname or #prefix/fieldname. When the id field contains a slash / the field name consists of the characters following the final slash (the prefix portion may contain one or more slashes to indicate scope). This defines a field of the wf step input object with the value of the source parameter(s).

Merging

To merge multiple inbound data links, MultipleInputFeatureRequirement must be specified in the wf or wf step requirements.

If the sink parameter is an array, or named in a wf scatter operation, there may be multiple inbound data links listed in the source field. The values from the input links are merged depending on the method specified in the linkMerge field. If not specified, the default method is “merge_nested”.

- **merge_nested** The input must be an array consisting of exactly one entry for each input link. If “merge_nested” is specified with a single link, the value from the link must be wrapped in a single-item list.
- **merge_flattened** 1. The source and sink parameters must be compatible types, or the source type must be compatible with single element from the “items” type of the destination array parameter. 2. Source parameters which are arrays are concatenated. Source parameters which are single element types are appended as single elements.

id

A unique identifier for this wf input parameter.

source

Specifies one or more wf parameters that will provide input to the underlying step parameter.

link_merge

The method to use to merge multiple inbound links into a single array. If not specified, the default method is “merge_nested”.

default

The default value for this parameter to use if either there is no source field, or the value produced by the source is null. The default must be applied prior to scattering or evaluating valueFrom.

value_from

To use valueFrom, StepInputExpressionRequirement must be specified in the wf or wf step requirements.

If valueFrom is a constant string value, use this as the value for this input parameter.

If valueFrom is a parameter reference or expression, it must be evaluated to yield the actual value to be assigned to the input field.

The self value of in the parameter reference or expression must be the value of the parameter(s) specified in the source field, or null if there is no source field.

The value of inputs in the parameter reference or expression must be the input object to the wf step after assigning the source values, applying default, and then scattering. The order of evaluating valueFrom among step input parameters is undefined and the result of evaluating valueFrom on a parameter must not be visible to evaluation of valueFrom on other parameters.

class `sbg.cwl.v1_0.StepOutput(id)`

Bases: `sbg.cwl.v1_0.base.Cwl`

Associate an output parameter of the underlying process with a wf parameter. The wf parameter (given in the id field) be may be used as a source to connect with input parameters of other wf steps, or with an output parameter of the process.

id

A unique identifier for this wf output parameter. This is the identifier to use in the source field of WorkflowStepInput to connect the output value to downstream parameters.

class `sbg.cwl.v1_0.Step(id, in_, out, run, requirements=None, hints=None, label=None, doc=None, scatter=None, scatter_method=None)`

Bases: `sbg.cwl.v1_0.base.Cwl`

A wf step is an executable element of a wf. 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 wf parameters. More on <http://www.commonwl.org/v1.0/Workflow.html#WorkflowStep>

link_merge (*id*, *method*='merge_nested')

Link merge port with *id* using *method*.

is_scattered (*k*)

Checks if port specified by *k* is scattered.

id

The unique identifier for this wf step.

in_

Defines the input parameters of the wf step. The process is ready to run when all required input parameters are associated with concrete values. Input parameters include a schema for each parameter which is used to validate the input object. It may also be used build a user interface for constructing the input object.

out

Defines the parameters representing the output of the process. May be used to generate and/or validate the output object.

run

Specifies the process to run.

requirements

Declares requirements that apply to either the runtime environment or the wf engine that must be met in order to execute this wf step. If an implementation cannot satisfy all requirements, or a requirement is listed which is not recognized by the implementation, it is a fatal error and the implementation must not attempt to run the process, unless overridden at user option.

hints

Declares hints applying to either the runtime environment or the wf engine that may be helpful in executing this wf step. It is not an error if an implementation cannot satisfy all hints, however the implementation may report a warning.

label

A short, human-readable label of this process object.

doc

A long, human-readable description of this process object.

scatter

The scatter field specifies one or more input parameters which will be scattered. An input parameter may be listed more than once. The declared type of each input parameter is implicitly becomes an array of items of the input parameter type. If a parameter is listed more than once, it becomes a nested array. As a result, upstream parameters which are connected to scattered parameters must be arrays.

scatter_method

Required if scatter is an array of more than one element.

```
class sbg.cwl.v1_0.ScatterMethod
```

Bases: object

The scatter method, as described in wf step scatter.

```
DOTPRODUCT = 'dotproduct'
```

```
NESTED_CROSSPRODUCT = 'nested_crossproduct'
```

```
FLAT_CROSSPRODUCT = 'flat_crossproduct'
```

```
class sbg.cwl.v1_0.Workflow(cwl_version='v1.0', inputs=None, outputs=None, steps=None,  
                           id=None, requirements=None, hints=None, label=None, doc=None,  
                           **kwargs)
```

Bases: sbg.cwl.v1_0.app.App

A wf 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.

When there is a dependency, the wf engine must execute the preceeding step and wait for it to successfully produce output before executing the dependent step. If two steps are defined in the wf graph that are not directly or indirectly dependent, these steps are independent, and may execute in any order or execute concurrently. A wf is complete when all steps have been executed.

Dependencies between parameters are expressed using the source field on wf step input parameters and wf output parameters.

The source field expresses the dependency of one parameter on another such that when a value is associated with the parameter specified by source, that value is propagated to the destination parameter. When all data links inbound to a given step are fulfilled, the step is ready to execute.

Workflow success and failure A completed step must result in one of success, temporaryFailure or permanentFailure states. An implementation may choose to retry a step execution which resulted in temporaryFailure. An implementation may choose to either continue running other steps of a wf, or terminate immediately upon permanentFailure.

- If any step of a wf execution results in permanentFailure, then the wf status is permanentFailure.
- If one or more steps result in temporaryFailure and all other steps complete success or are not executed, then the wf status is temporaryFailure.
- If all wf steps are executed and complete with success, then the wf status is success.

class_ = 'Workflow'

get_requirements (*value*)

get_step (*id*)

Get step by id.

add_requirement (*new_r*)

Adds *new_r* into list of workflow requirements.

add_connection (*src*, *dst*)

Connects source and destination nodes, specified by *src* and *dst* ids respectively.

Parameters

- **src** – connection source
- **dst** – connection destination

scatter (*step*, *ports*, *method*)

Scatter input ports on *step*.

Parameters

- **step** – step on which scatter is performed
- **ports** – step ports
- **method** – scatter method, required when len(ports) > 1

add_step (*step*, *id=None*, *in_=None*, *out=None*, *expose=None*, *expose_except=None*, *scatter=None*, *scatter_method=None*)

Adds step into workflow.

Parameters

- **step** – can be either instance of App subclass or WorkflowStep
- **id** – step id

- **in** – step inputs
- **out** – step outputs
- **expose** – list or map of keys to be exposed as workflow IO (default expose everything)
- **expose_except** – list of ports to be excluded from exposing
- **scatter** – scatter inputs
- **scatter_method** – scattering method required when `scatter` is a list with > 1 port

steps

The individual steps that make up the wf. Each step is executed when all of its input data links are fulfilled. An implementation may choose to execute the steps in a different order than listed and/or execute steps concurrently, provided that dependencies between steps are met.

```
class sbg.cwl.v1_0.SubworkflowFeature
```

Bases: `sbg.cwl.v1_0.base.Cwl`

Indicates that the wf platform must support nested workflows in the `run` field of `WorkflowStep`.

```
class_ = 'SubworkflowFeatureRequirement'
```

```
class sbg.cwl.v1_0.ScatterFeature
```

Bases: `sbg.cwl.v1_0.base.Cwl`

Indicates that the wf platform must support the `scatter` and `scatterMethod` fields of `WorkflowStep`.

```
class_ = 'ScatterFeatureRequirement'
```

```
class sbg.cwl.v1_0.MultipleInputFeature
```

Bases: `sbg.cwl.v1_0.base.Cwl`

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

```
class_ = 'MultipleInputFeatureRequirement'
```

```
class sbg.cwl.v1_0.StepInputExpression
```

Bases: `sbg.cwl.v1_0.base.Cwl`

Indicate that the wf platform must support the `valueFrom` field of `WorkflowStepInput`.

```
class_ = 'StepInputExpressionRequirement'
```

```
class sbg.cwl.v1_0.ExpressionTool(expression, cwl_version='v1.0', inputs=None, outputs=None, id=None, requirements=None, hints=None, label=None, doc=None, **kwargs)
```

Bases: `sbg.cwl.v1_0.app.App`

Execute an expression as a `Workflow` step.

```
class_ = 'ExpressionTool'
```

```
get_requirements (value)
```

```
expression
```

A long, human-readable description of this process object.

```
class sbg.cwl.v1_0.InputBinding(load_contents=None, position=None, prefix=None, separate=None, item_separator=None, value_from=None, shell_quote=None)
```

Bases: `sbg.cwl.v1_0.base.Cwl`

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`.

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.

- **string**: Add prefix and the string to the command line.
- **number**: Add prefix and decimal representation to command line.
- **boolean**: If true, add prefix to the command line. If false, add nothing.
- **File**: Add prefix and the value of `File.path` to the command line.
- **array**: If `itemSeparator` is specified, add prefix and the join the array into a single string with `itemSeparator` separating the items. Otherwise first add prefix, then recursively process individual elements.
- **object**: Add prefix only, and recursively add object fields for which `inputBinding` is specified.
- **null**: Add nothing.

load_contents

Only valid when type `File` or is an array of items `File`. Read up to the first 64 KiB of text from the file and place it in the “contents” field of the file object for use by expressions.

position

The sorting key. Default position is 0.

prefix

Command line prefix to add before the value.

separate

If true (default), then the prefix and value must be added as separate command line arguments; if false, prefix and value must be concatenated into a single command line argument.

item_separator

Join the array elements into a single string with the elements separated by `itemSeparator`.

value_from

If `valueFrom` is a constant string value, use this as the value and apply the binding rules above.

If `valueFrom` is an expression, evaluate the expression to yield the actual value to use to build the command line and apply the binding rules above. If the `inputBinding` is associated with an input parameter, the value of `self` in the expression will be the value of the input parameter. Input parameter defaults (as specified by the `InputParameter` default field) must be applied before evaluating the expression.

When a binding is part of the `CommandLineTool.arguments` field, the `valueFrom` field is required.

shell_quote

If `ShellCommandRequirement` is in the requirement for the current command, this controls whether the value is quoted on the command line (default is true). Use `shellQuote`: false to inject metacharacters for operations such as pipes.

If `shellQuote` is true or not provided, the implementation must not permit interpretation of any shell metacharacters or directives.

```
class sbg.cwl.v1_0.InputRecordField(name, type, doc=None, input_binding=None, label=None)
```

Bases: `sbg.cwl.v1_0.schema.RecordFieldBase`

input_binding

get_type (value)

```
class sbg.cwl.v1_0.InputRecord (label=None, fields=None, type='record')
    Bases: sbg.cwl.v1_0.schema.RecordBase

    get_field_cls()

class sbg.cwl.v1_0.InputEnum (symbols, label=None, input_binding=None, type='enum')
    Bases: sbg.cwl.v1_0.schema.EnumBase

    input_binding

class sbg.cwl.v1_0.InputArray (items, label=None, input_binding=None, type='array')
    Bases: sbg.cwl.v1_0.schema.ArrayBase

    input_binding

    get_items (value)

class sbg.cwl.v1_0.OutputRecord (label=None, fields=None, type='record')
    Bases: sbg.cwl.v1_0.schema.RecordBase

    get_field_cls()

class sbg.cwl.v1_0.OutputRecordField (name, type, doc=None, output_binding=None)
    Bases: sbg.cwl.v1_0.schema.RecordFieldBase

    output_binding

    get_type (value)

class sbg.cwl.v1_0.OutputEnum (symbols, label=None, output_binding=None, type='enum')
    Bases: sbg.cwl.v1_0.schema.EnumBase

    output_binding

class sbg.cwl.v1_0.OutputArray (items, label=None, output_binding=None, type='array')
    Bases: sbg.cwl.v1_0.schema.ArrayBase

    output_binding

    get_items (value)

class sbg.cwl.v1_0.OutputBinding (glob=None, load_contents=None, output_eval=None)
    Bases: sbg.cwl.v1_0.base.Cwl
```

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

The output parameter value is generated by applying these operations in the following order:

- glob
- loadContents
- outputEval
- secondaryFiles

glob

Find files relative to the output directory, using POSIX glob(3) pathname matching. If an array is provided, find files that match any pattern in the array. If an expression is provided, the expression must return a string or an array of strings, which will then be evaluated as one or more glob patterns. Must only match and return files which actually exist.

load_contents

For each file matched in glob, read up to the first 64 KiB of text from the file and place it in the contents field of the file object for manipulation by outputEval.

output_eval

Evaluate an expression to generate the output value. If glob was specified, the value of self must be an array containing file objects that were matched. If no files were matched, self must be a zero length array; if a single file was matched, the value of self is an array of a single element. Additionally, if loadContents is true, the File objects must include up to the first 64 KiB of file contents in the contents field.

```
class sbg.cwl.v1_0.App (class_, cwl_version='v1.0', inputs=None, outputs=None, id=None, require-  
ments=None, hints=None, label=None, doc=None)
```

Bases: sbg.cwl.v1_0.base.Cwl

Super class for all runnable objects:

- CommandLineTool
- ExpressionTool
- Workflow

add_hints (*hints)

Adds application hints.

add_input_json ()

Creates and adds input.json as workdir requirement. File input.json contains all information about inputs in runtime.

find_requirement (name)

Returns requirement by class name.

get_port (id)

Returns input/output port specified by id.

Parameters id – IO unique identifier

Returns IO port

get_input (id)

Returns input by its id.

Parameters id – input id

Returns input object

get_output (id)

Returns output by its id.

Parameters id – output id

Returns output object

create_file (entry, entryname=None, writable=None, encode=False)

Returns created file (Dirent).

Parameters

- **entry** – file content
- **entryname** – file name
- **writable** – flag that indicates that file is writable
- **encode** – encode entry using base64

Returns an instance of Dirent

add_file (entry, entryname=None, writable=None, encode=False)

Creates file (Dirent) and adds it into InitialWorkDirRequirement.

Parameters

- **entry** – file content
- **entryname** – file name
- **writable** – flag that indicates that file is writable
- **encode** – encode entry using `base64`

stage_input (*id*)Stages input file specified by its `id` into current working dir.**Parameters** **id** – input file id**set_secondary_files** (*id, secondary*)Sets secondary files on input/output identified with *id*.**Parameters**

- **id** – object id
- **secondary** – provides a pattern or expression specifying files or directories that must be included alongside the primary file

inherit_metadata (*src, dst*)Inherits metadata from `src` input id to `dst` output id.**Parameters**

- **src** – input ID
- **dst** – output ID

static add_sbg_namespace (*app*)

Adds SBG namespace for CWL application (tool, workflow).

Parameters **app** – an instance of `cwl.App`**Returns** `app` with added SBG namespace**static set_required** (*obj, required*)If argument `required=True` return required object. If argument `required=False` return non required object.**Parameters**

- **obj** – type object
- **required** – flag

Returns type object**static is_required** (*obj*)Checks if `obj` is required type.**Parameters** **obj** – type object**Returns** `bool`**add_expression_lib** (*lib*)Adds expression library into `InlineJavascriptRequirement`.**Parameters** **lib** – javascript library (can be string or list of strings)**add_env_var** (*env_name=None, env_value=None, env_def=None*)

Adds environment variable into requirements.

Parameters

- **env_name** – the environment variable name
- **env_value** – the environment variable value
- **env_def** – the list of environment variables

add_in_workdir (*r*)Adds *r* into InitialWorkDirRequirement.

Parameters *r* – the list of files or subdirectories that must be placed in the designated output directory prior to executing the command line tool

add_requirement (*new_r*)Add *new_r* into requirements.**add_input** (*hint*, *id=None*, *label=None*, *secondary_files=None*, *streamable=None*, *doc=None*, *format=None*, *input_binding=None*, *stage=False*)

Adds input by type hint and kwargs arguments.

Parameters

- **hint** – type hint (eg, `Int()`, `String()`, etc)
- **id** – the unique identifier for this parameter object
- **label** – a short, human-readable label of this object
- **secondary_files** – a list of additional files or directories that are associated with the primary file and must be transferred alongside the primary file
- **streamable** – a value of true indicates that the file is read or written sequentially without seeking
- **doc** – a documentation string for this type, or an array of strings which should be concatenated
- **format** – the format of the file
- **input_binding** – describes how to handle the inputs of a process and convert them into a concrete form for execution, such as command line parameters.
- **stage** – stages input file into current working dir

add_output (*hint*, *id=None*, *label=None*, *secondary_files=None*, *streamable=None*, *doc=None*, *output_binding=None*, *format=None*)

Adds output by type hint and kwargs arguments.

Parameters

- **id** – the unique identifier for this parameter object
- **label** – a short, human-readable label of this object
- **secondary_files** – provides a pattern or expression specifying files or directories that must be included alongside the primary file.
- **streamable** – a value of true indicates that the file is read or written sequentially without seeking
- **doc** – a documentation string for this type, or an array of strings which should be concatenated
- **output_binding** – describes how to handle the outputs of a process
- **format** – the format of the file

inputs

Defines the input parameters of the process. The process is ready to run when all required input parameters are associated with concrete values. Input parameters include a schema for each parameter which is used to validate the input object. It may also be used to build a user interface for constructing the input object.

When accepting an input object, all input parameters must have a value. If an input parameter is missing from the input object, it must be assigned a value of null (or the value of default for that parameter, if provided) for the purposes of validation and evaluation of expressions.

outputs

Defines the parameters representing the output of the process. May be used to generate and/or validate the output object.

id

The unique identifier for this process object.

get_requirements (*value*)**requirements**

Declares requirement that apply to either the runtime environment or the wf engine that must be met in order to execute this process. If an implementation cannot satisfy all requirement, or a requirement is listed which is not recognized by the implementation, it is a fatal error and the implementation must not attempt to run the process, unless overridden at user option.

hints

Declares hints applying to either the runtime environment or the wf engine that may be helpful in executing this process. It is not an error if an implementation cannot satisfy all hints, however the implementation may report a warning.

label

A short, human-readable label of this process object.

doc

A long, human-readable description of this process object.

cwl_version

CWL document version. Always required at the document root. Not required for a Process embedded inside another Process.

```
class sbg.cwl.v1_0.Int (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
                        quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
```

type

```
class sbg.cwl.v1_0.Float (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
                        quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
```

type

```
class sbg.cwl.v1_0.Bool (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
                        quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
```

type

```
class sbg.cwl.v1_0.String (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None,
                        required=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
```

type

```
class sbg.cwl.v1_0.Any (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
    quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
    type

class sbg.cwl.v1_0.Array (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
    quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
    type

class sbg.cwl.v1_0.Enum (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
    quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
    type

class sbg.cwl.v1_0.Record (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None,
    required=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
    type

class sbg.cwl.v1_0.File (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
    quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
    type

class sbg.cwl.v1_0.Dir (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
    quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
    type

class sbg.cwl.v1_0.Union (val=None, default=<class 'sbg.cwl.v1_0.hints.Empty'>, glob=None, re-
    quired=False, label=None, doc=None, secondary_files=None)
    Bases: sbg.cwl.v1_0.hints.Hint
    type
```

4.1 Submodules

4.2 sbg.cwl.sbg.session module

```
class sbg.cwl.sbg.session.Session (profile='default', endpoint=None, token=None, api=None)
    Bases: object
```

Session for running CWL documents on SBG platform.

Parameters

- **profile** – profile listed inside `$HOME/.sevenbridges/credentials` (eg. `cgc`, `default`, `cavatica`)
- **endpoint** – api endpoint
- **token** – developer token from platform
- **api** – sbg api object

Example:

```
from sbg import cwl

session = cwl.Session() # default will be used
session = cwl.Session(token='<DEV_TOKEN>', endpoint='<ENDPOINT>')
```

static init_api (*profile='default', token=None, endpoint=None*)

Initialize SBG API using credentials located inside \$HOME/.sevenbridges/credentials or by provided dev_token and platform.

Parameters

- **profile** – profile listed inside \$HOME/.sevenbridges/credentials. Example: cgc, default, cavatica
- **token** – developer token from platform
- **endpoint** – api endpoint

Returns an instance of Api

create_app (*app, project*)

Install/create revision of app. New revision is created only if there is a difference from a latest revision.

Parameters

- **app** – an instance of cwl.App
- **project** – an instance of Project

Returns installed app

Example:

```
from sbg import cwl

session = cwl.Session()
session.create_app(
    app=cwl.CommandLineTool(id='my_id'),
    project='<PROJECT>'
)
```

draft (*project, app, inputs=None, hints=None*)

Creates draft task.

Parameters

- **project** – an instance of either Project or str
- **app** – an instance of cwl.App
- **inputs** – input map
- **hints** – SBG hints

Returns created draft task

Example:

```
from sbg import cwl

session = cwl.Session()
session.draft(
    app=cwl.CommandLineTool(
```

(continues on next page)

(continued from previous page)

```
        id='my_id',
        base_command=['echo', 'SevenBridges']
    ),
    project='<PROJECT>'
)
```

static add_hints (*app*, **hints*)

Add hints on a app.

Parameters

- **app** – an instance of `cwl.App`
- **hints** – hints

Returns app with added hints**run** (*project*, *app*, *inputs=None*, *hints=None*)

Runs app on a platform.

Parameters

- **project** – an instance of either `Project` or `str`
- **app** – an instance of `cwl.App`
- **inputs** – input map
- **hints** – list of `Hint`

Returns task

Example:

```
from sbg import cwl

session = cwl.Session()
session.run(
    app=cwl.CommandLineTool(
        id='my_id',
        base_command=['echo', 'SevenBridges']
    ),
    project='<PROJECT>'
)
```

4.3 sbg.cwl.v1_0.schema module

`sbg.cwl.v1_0.schema.input_schema_item` (*value*)

```
class sbg.cwl.v1_0.schema.InputBinding (load_contents=None, position=None, pre-
                                         fix=None, separate=None, item_separator=None,
                                         value_from=None, shell_quote=None)
```

Bases: `sbg.cwl.v1_0.base.Cwl`

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`.

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.

- **string:** Add prefix and the string to the command line.
- **number:** Add prefix and decimal representation to command line.
- **boolean:** If true, add prefix to the command line. If false, add nothing.
- **File:** Add prefix and the value of File.path to the command line.
- **array:** If itemSeparator is specified, add prefix and the join the array into a single string with itemSeparator separating the items. Otherwise first add prefix, then recursively process individual elements.
- **object:** Add prefix only, and recursively add object fields for which inputBinding is specified.
- **null:** Add nothing.

load_contents

Only valid when type File or is an array of items File. Read up to the first 64 KiB of text from the file and place it in the “contents” field of the file object for use by expressions.

position

The sorting key. Default position is 0.

prefix

Command line prefix to add before the value.

separate

If true (default), then the prefix and value must be added as separate command line arguments; if false, prefix and value must be concatenated into a single command line argument.

item_separator

Join the array elements into a single string with the elements separated by itemSeparator.

value_from

If valueFrom is a constant string value, use this as the value and apply the binding rules above.

If valueFrom is an expression, evaluate the expression to yield the actual value to use to build the command line and apply the binding rules above. If the inputBinding is associated with an input parameter, the value of self in the expression will be the value of the input parameter. Input parameter defaults (as specified by the InputParameter default field) must be applied before evaluating the expression.

When a binding is part of the CommandLineTool.arguments field, the valueFrom field is required.

shell_quote

If ShellCommandRequirement is in the requirement for the current command, this controls whether the value is quoted on the command line (default is true). Use shellQuote: false to inject metacharacters for operations such as pipes.

If shellQuote is true or not provided, the implementation must not permit interpretation of any shell metacharacters or directives.

```
class sbg.cwl.v1_0.schema.OutputBinding(glob=None, load_contents=None, out-  
put_eval=None)
```

Bases: sbg.cwl.v1_0.base.Cwl

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

The output parameter value is generated by applying these operations in the following order:

- glob
- loadContents

- `outputEval`
- `secondaryFiles`

glob

Find files relative to the output directory, using POSIX `glob(3)` pathname matching. If an array is provided, find files that match any pattern in the array. If an expression is provided, the expression must return a string or an array of strings, which will then be evaluated as one or more glob patterns. Must only match and return files which actually exist.

load_contents

For each file matched in `glob`, read up to the first 64 KiB of text from the file and place it in the `contents` field of the file object for manipulation by `outputEval`.

output_eval

Evaluate an expression to generate the output value. If `glob` was specified, the value of `self` must be an array containing file objects that were matched. If no files were matched, `self` must be a zero length array; if a single file was matched, the value of `self` is an array of a single element. Additionally, if `loadContents` is true, the File objects must include up to the first 64 KiB of file contents in the `contents` field.

```
class sbg.cwl.v1_0.schema.SchemaBase
```

```
    Bases: sbg.cwl.v1_0.base.Cwl
```

```
class sbg.cwl.v1_0.schema.UnionBase
```

```
    Bases: sbg.cwl.v1_0.schema.SchemaBase
```

```
class sbg.cwl.v1_0.schema.RecordBase (label=None, fields=None, type='record')
```

```
    Bases: sbg.cwl.v1_0.schema.SchemaBase
```

```
    type = 'record'
```

```
    fields
```

```
        Defines the fields of the record.
```

```
    label
```

```
        A short, human-readable label of this object.
```

```
    to_input ()
```

```
    to_output ()
```

```
    get_fields (**kwargs)
```

```
    get_field_cls ()
```

```
class sbg.cwl.v1_0.schema.InputRecord (label=None, fields=None, type='record')
```

```
    Bases: sbg.cwl.v1_0.schema.RecordBase
```

```
    get_field_cls ()
```

```
class sbg.cwl.v1_0.schema.OutputRecord (label=None, fields=None, type='record')
```

```
    Bases: sbg.cwl.v1_0.schema.RecordBase
```

```
    get_field_cls ()
```

```
class sbg.cwl.v1_0.schema.RecordFieldBase (name, type, doc=None)
```

```
    Bases: sbg.cwl.v1_0.schema.SchemaBase
```

```
    name
```

```
        The name of the field.
```

```
    type
```

```
        The field type
```

```
    doc
        A documentation string for this field

    to_input ()

    to_output ()

    get_type (value)

class sbg.cwl.v1_0.schema.InputRecordField (name, type, doc=None, input_binding=None,
                                             label=None)
    Bases: sbg.cwl.v1_0.schema.RecordFieldBase
    input_binding
    get_type (value)

class sbg.cwl.v1_0.schema.OutputRecordField (name, type, doc=None, out-
                                             put_binding=None)
    Bases: sbg.cwl.v1_0.schema.RecordFieldBase
    output_binding
    get_type (value)

class sbg.cwl.v1_0.schema.ArrayBase (items, label, type)
    Bases: sbg.cwl.v1_0.schema.SchemaBase
    type = 'array'
    items_
        Defines the type of the array elements.
    label
        A short, human-readable label of this object.
    to_input ()
    to_output ()
    get_items (value)

class sbg.cwl.v1_0.schema.InputArray (items, label=None, input_binding=None,
                                       type='array')
    Bases: sbg.cwl.v1_0.schema.ArrayBase
    input_binding
    get_items (value)

class sbg.cwl.v1_0.schema.OutputArray (items, label=None, output_binding=None,
                                       type='array')
    Bases: sbg.cwl.v1_0.schema.ArrayBase
    output_binding
    get_items (value)

class sbg.cwl.v1_0.schema.EnumBase (symbols, label, type)
    Bases: sbg.cwl.v1_0.schema.SchemaBase
    type = 'enum'
    symbols
        Defines the set of valid symbols.
    label
        A short, human-readable label of this object.
```

```
to_input()
to_output()
class sbg.cwl.v1_0.schema.InputEnum(symbols, label=None, input_binding=None,
                                     type='enum')
    Bases: sbg.cwl.v1_0.schema.EnumBase
    input_binding
class sbg.cwl.v1_0.schema.OutputEnum(symbols, label=None, output_binding=None,
                                     type='enum')
    Bases: sbg.cwl.v1_0.schema.EnumBase
    output_binding
sbg.cwl.v1_0.schema.set_required(obj, required)
    If argument required=True return required object. If argument required=False return non required
    object.
    Parameters
        • obj – type object
        • required – flag
    Returns type object
```

4.4 sbg.cwl.v1_0.cmd module

```
class sbg.cwl.v1_0.cmd.CommandLineTool(cwl_version='v1.0', inputs=None, out-
                                     puts=None, id=None, requirements=None,
                                     hints=None, label=None, doc=None,
                                     base_command=None, arguments=None,
                                     stdin=None, stdout=None, stderr=None, suc-
                                     cess_codes=None, temporary_fail_codes=None,
                                     permanent_fail_codes=None, **kwargs)
```

Bases: sbg.cwl.v1_0.app.App

This defines the schema of the CWL Command Line Tool Description document.

```
class_ = 'CommandLineTool'
```

```
unarchive_bundle(bundle, encoded=False, postprocess=None)
```

Adds first unarchiving command as first argument. It can also decode base64 encoded bundles.

Parameters

- **bundle** – bundle name
- **encoded** – flag which indicates that bundle is base64 encoded

```
static get_unarchive_argument(bundle, encoded=False, postprocess=None)
```

Returns InputBinding argument with untar command.

Parameters

- **bundle** – archive name
- **encoded** – flag which indicates that bundle is base64 encoded

Returns an instance of InputBinding which is an argument

add_locals (*locals, name, postprocess=None*)

Add local files/dirs to tool in runtime.

Parameters

- **locals** – list with paths
- **name** – bundle name in runtime
- **postprocess** – bash operation after unarchiving a bundle

classmethod from_bash (*script, name='script.sh', id=None, label=None, doc=None, inputs=None, outputs=None, sources=None, lib=None, docker=None, secondary_files=None, stdout=None*)

Creates CommandLineTool created from bash script.

Parameters

- **script** – can be bash file or contents
- **name** – bash script name
- **id** – tool ID
- **label** – tool label
- **doc** – tool description
- **inputs** – dictionary where keys are variable names and values can be either one of cwl hints or string. if hint is specified, then corresponding input will be created. If string is specified, then environment variable with that string value will be created.
- **outputs** – dictionary where keys are output ids and values are cwl hints
- **sources** – executes `source <source>` which can be used for exporting bash variables
- **lib** – only required when sources are specified, used as a name for bundle with sources
- **docker** – specify a docker image to retrieve using docker pull
- **secondary_files** – dictionary where key is input/output id and value is secondary files
- **stdout** – standard output redirect to this file

Returns an instance of `cwl.CommandLineTool`

get_requirements (*value*)

base_command

Specifies the program to execute. If an array, the first element of the array is the command to execute, and subsequent elements are mandatory command line arguments. The elements in `baseCommand` must appear before any command line bindings from `inputBinding` or arguments.

If `baseCommand` is not provided or is an empty array, the first element of the command line produced after processing `inputBinding` or arguments must be used as the program to execute.

If the program includes a path separator character it must be an absolute path, otherwise it is an error. If the program does not include a path separator, search the `$PATH` variable in the runtime environment of the wf runner find the absolute path of the executable.

arguments

Command line bindings which are not directly associated with input parameters.

stdin

A path to a file whose contents must be piped into the command's standard input stream.

stdout

Capture the command's standard output stream to a file written to the designated output directory.

If stdout is a string, it specifies the file name to use.

If stdout is an expression, the expression is evaluated and must return a string with the file name to use to capture stdout. If the return value is not a string, or the resulting path contains illegal characters (such as the path separator /) it is an error.

stderr

Capture the command's standard error stream to a file written to the designated output directory.

If stderr is a string, it specifies the file name to use.

If stderr is an expression, the expression is evaluated and must return a string with the file name to use to capture stderr. If the return value is not a string, or the resulting path contains illegal characters (such as the path separator /) it is an error.

success_codes

Exit codes that indicate the process completed successfully.

temporary_fail_codes

Exit codes that indicate the process failed due to a possibly temporary condition, where executing the process with the same runtime environment and inputs may produce different results.

permanent_fail_codes

Exit codes that indicate the process failed due to a permanent logic error, where executing the process with the same runtime environment and same inputs is expected to always fail.

```
class sbg.cwl.v1_0.cmd.CommandInput (id=None, label=None, secondary_files=None,
                                     streamable=None, doc=None, format=None, input_binding=None, default=None, type=None)
```

Bases: sbg.cwl.v1_0.base.Cwl

An input parameter for a CommandLineTool.

id

The unique identifier for this parameter object.

label

A short, human-readable label of this object.

secondary_files

Only valid when type – File or is an array of items File.

Provides a pattern or expression specifying files or directories that must be included alongside the primary file. All listed secondary files must be present. An implementation may fail wf execution if an expected secondary file does not exist.

If the value is an expression, the value of self in the expression must be the primary input or output File object to which this binding applies. The basename, nameroot and nameext fields must be present in self. For CommandLineTool outputs the path field must also be present. The expression must return a filename string relative to the path to the primary File, a File or Directory object with either path or location and basename fields set, or an array consisting of strings or File or Directory objects. It is legal to reference an unchanged File or Directory object taken from input as a secondaryFile.

To work on non-filename-preserving storage systems, portable tool descriptions should avoid constructing new values from location, but should construct relative references using basename or nameroot instead.

If a value in secondaryFiles is a string that is not an expression, it specifies that the following pattern should be applied to the path of the primary file to yield a filename relative to the primary File:

- If string begins with one or more caret ^ characters, for each caret, remove the last file extension from the path (the last period . and all following characters). If there are no file extensions, the path is unchanged.
- Append the remainder of the string to the end of the file path.

streamable

Only valid when type File or is an array of items File.

A value of true indicates that the file is read or written sequentially without seeking. An implementation may use this flag to indicate whether it is valid to stream file contents using a named pipe. Default: false.

doc

A documentation string for this type, or an array of strings which should be concatenated.

format

Only valid when type File or is an array of items File.

This must be one or more IRIs of concept nodes that represents file formats which are allowed as input to this parameter, preferably defined within an ontology. If no ontology is available, file formats may be tested by exact match.

input_binding

Describes how to handle the inputs of a process and convert them into a concrete form for execution, such as command line parameters.

default

The default value to use for this parameter if the parameter is missing from the input object, or if the value of the parameter in the input object is null. Default values are applied before evaluating expressions (e.g. dependent valueFrom fields).

type

Specify valid types of data that may be assigned to this parameter.

```
class sbg.cwl.v1_0.cmd.CommandOutput (id=None, label=None, secondary_files=None, streamable=None, doc=None, output_binding=None, format=None, type=None)
```

Bases: sbg.cwl.v1_0.base.Cwl

An output parameter for a CommandLineTool.

id

The unique identifier for this parameter object.

label

A short, human-readable label of this object.

secondary_files

Only valid when type File or is an array of items File.

Provides a pattern or expression specifying files or directories that must be included alongside the primary file. All listed secondary files must be present. An implementation may fail wf execution if an expected secondary file does not exist.

If the value is an expression, the value of self in the expression must be the primary input or output File object to which this binding applies. The basename, nameroot and nameext fields must be present in self. For CommandLineTool outputs the path field must also be present. The expression must return a filename string relative to the path to the primary File, a File or Directory object with either path or location and basename fields set, or an array consisting of strings or File or Directory objects. It is legal to reference an unchanged File or Directory object taken from input as a secondaryFile.

To work on non-filename-preserving storage systems, portable tool descriptions should avoid constructing new values from location, but should construct relative references using basename or nameroot instead.

If a value in `secondaryFiles` is a string that is not an expression, it specifies that the following pattern should be applied to the path of the primary file to yield a filename relative to the primary File

1. If string begins with one or more caret `^` characters, for each caret, remove the last file extension from the path (the last period `.` and all following characters). If there are no file extensions, the path is unchanged.
2. Append the remainder of the string to the end of the file path.

streamable

Only valid when type File or is an array of items File.

A value of true indicates that the file is read or written sequentially without seeking. An implementation may use this flag to indicate whether it is valid to stream file contents using a named pipe. Default false.

doc

A documentation string for this type, or an array of strings which should be concatenated.

output_binding

Describes how to handle the outputs of a process.

format

Only valid when type File or is an array of items File.

This is the file format that will be assigned to the output parameter.

type

Specify valid types of data that may be assigned to this parameter.

4.5 sbg.cwl.v1_0.wf module

```
class sbg.cwl.v1_0.wf.Workflow(cwl_version='v1.0', inputs=None, outputs=None, steps=None,
                                id=None, requirements=None, hints=None, label=None,
                                doc=None, **kwargs)
```

Bases: `sbg.cwl.v1_0.app.App`

A wf 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.

When there is a dependency, the wf engine must execute the preceeding step and wait for it to successfully produce output before executing the dependent step. If two steps are defined in the wf graph that are not directly or indirectly dependent, these steps are independent, and may execute in any order or execute concurrently. A wf is complete when all steps have been executed.

Dependencies between parameters are expressed using the source field on wf step input parameters and wf output parameters.

The source field expresses the dependency of one parameter on another such that when a value is associated with the parameter specified by source, that value is propagated to the destination parameter. When all data links inbound to a given step are fulfilled, the step is ready to execute.

Workflow success and failure A completed step must result in one of success, `temporaryFailure` or `permanentFailure` states. An implementation may choose to retry a step execution which resulted in `temporaryFailure`. An implementation may choose to either continue running other steps of a wf, or terminate immediately upon `permanentFailure`.

- If any step of a wf execution results in `permanentFailure`, then the wf status is `permanentFailure`.
- If one or more steps result in `temporaryFailure` and all other steps complete success or are not executed, then the wf status is `temporaryFailure`.
- If all wf steps are executed and complete with success, then the wf status is success.

```
class_ = 'Workflow'
```

```
get_requirements (value)
```

```
get_step (id)
```

Get step by *id*.

```
add_requirement (new_r)
```

Adds *new_r* into list of workflow requirements.

```
add_connection (src, dst)
```

Connects source and destination nodes, specified by *src* and *dst* ids respectively.

Parameters

- **src** – connection source
- **dst** – connection destination

```
scatter (step, ports, method)
```

Scatter input ports on *step*.

Parameters

- **step** – step on which scatter is performed
- **ports** – step ports
- **method** – scatter method, required when `len(ports) > 1`

```
add_step (step, id=None, in=None, out=None, expose=None, expose_except=None, scatter=None,
          scatter_method=None)
```

Adds step into workflow.

Parameters

- **step** – can be either instance of App subclass or WorkflowStep
- **id** – step id
- **in** – step inputs
- **out** – step outputs
- **expose** – list or map of keys to be exposed as workflow IO (default expose everything)
- **expose_except** – list of ports to be excluded from exposing
- **scatter** – scatter inputs
- **scatter_method** – scattering method required when *scatter* is a list with > 1 port

steps

The individual steps that make up the wf. Each step is executed when all of its input data links are fulfilled. An implementation may choose to execute the steps in a different order than listed and/or execute steps concurrently, provided that dependencies between steps are met.

```
class sbg.cwl.v1_0.wf.WorkflowOutput (id=None, label=None, secondary_files=None, streamable=None,
                                     doc=None, output_binding=None,
                                     format=None, type=None, output_source=None,
                                     link_merge=None)
```

Bases: `sbg.cwl.v1_0.cmd.output.CommandOutput`

```
output_source
```

Specifies one or more wf parameters that supply the value of to the output parameter.

link_merge

The method to use to merge multiple sources into a single array. If not specified, the default method is “merge_nested”.

```
class sbg.cwl.v1_0.wf.WorkflowInput(id=None, label=None, secondary_files=None,  
                                   streamable=None, doc=None, format=None, in-  
                                   put_binding=None, default=None, type=None)  
    Bases: sbg.cwl.v1_0.cmd.input.CommandInput
```

```
class sbg.cwl.v1_0.wf.MergeMethod
```

Bases: object

MERGE_NESTED = 'merge_nested'

MERGE_FLATTENED = 'merge_flattened'

```
class sbg.cwl.v1_0.wf.StepInputExpression
```

Bases: sbg.cwl.v1_0.base.Cwl

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

class_ = 'StepInputExpressionRequirement'

```
class sbg.cwl.v1_0.wf.MultipleInputFeature
```

Bases: sbg.cwl.v1_0.base.Cwl

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

class_ = 'MultipleInputFeatureRequirement'

```
class sbg.cwl.v1_0.wf.ScatterFeature
```

Bases: sbg.cwl.v1_0.base.Cwl

Indicates that the wf platform must support the scatter and scatterMethod fields of WorkflowStep.

class_ = 'ScatterFeatureRequirement'

```
class sbg.cwl.v1_0.wf.SubworkflowFeature
```

Bases: sbg.cwl.v1_0.base.Cwl

Indicates that the wf platform must support nested workflows in the run field of WorkflowStep.

class_ = 'SubworkflowFeatureRequirement'

```
class sbg.cwl.v1_0.wf.ScatterMethod
```

Bases: object

The scatter method, as described in wf step scatter.

DOTPRODUCT = 'dotproduct'

NESTED_CROSSPRODUCT = 'nested_crossproduct'

FLAT_CROSSPRODUCT = 'flat_crossproduct'

```
class sbg.cwl.v1_0.wf.Step(id, in_, out, run, requirements=None, hints=None, label=None,  
                          doc=None, scatter=None, scatter_method=None)  
    Bases: sbg.cwl.v1_0.base.Cwl
```

A wf step is an executable element of a wf. 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 wf parameters. More on <http://www.commonwl.org/v1.0/Workflow.html#WorkflowStep>

link_merge (*id, method='merge_nested'*)

Link merge port with *id* using *method*.

is_scattered (*k*)

Checks if port specified by *k* is scattered.

id

The unique identifier for this wf step.

in_

Defines the input parameters of the wf step. The process is ready to run when all required input parameters are associated with concrete values. Input parameters include a schema for each parameter which is used to validate the input object. It may also be used build a user interface for constructing the input object.

out

Defines the parameters representing the output of the process. May be used to generate and/or validate the output object.

run

Specifies the process to run.

requirements

Declares requirements that apply to either the runtime environment or the wf engine that must be met in order to execute this wf step. If an implementation cannot satisfy all requirements, or a requirement is listed which is not recognized by the implementation, it is a fatal error and the implementation must not attempt to run the process, unless overridden at user option.

hints

Declares hints applying to either the runtime environment or the wf engine that may be helpful in executing this wf step. It is not an error if an implementation cannot satisfy all hints, however the implementation may report a warning.

label

A short, human-readable label of this process object.

doc

A long, human-readable description of this process object.

scatter

The scatter field specifies one or more input parameters which will be scattered. An input parameter may be listed more than once. The declared type of each input parameter is implicitly becomes an array of items of the input parameter type. If a parameter is listed more than once, it becomes a nested array. As a result, upstream parameters which are connected to scattered parameters must be arrays.

scatter_method

Required if scatter is an array of more than one element.

class `sbg.cwl.v1_0.wf.StepOutput` (*id*)

Bases: `sbg.cwl.v1_0.base.Cwl`

Associate an output parameter of the underlying process with a wf parameter. The wf parameter (given in the *id* field) be may be used as a source to connect with input parameters of other wf steps, or with an output parameter of the process.

id

A unique identifier for this wf output parameter. This is the identifier to use in the source field of WorkflowStepInput to connect the output value to downstream parameters.

class `sbg.cwl.v1_0.wf.StepInput` (*id*, *source=None*, *link_merge=None*, *default=None*,
value_from=None)

Bases: `sbg.cwl.v1_0.base.Cwl`

The input of a wf step connects an upstream parameter (from the wf inputs, or the outputs of other workflows steps) with the input parameters of the underlying step.

Input object

A WorkflowStepInput object must contain an id field in the form #fieldname or #prefix/fieldname. When the id field contains a slash / the field name consists of the characters following the final slash (the prefix portion may contain one or more slashes to indicate scope). This defines a field of the wf step input object with the value of the source parameter(s).

Merging

To merge multiple inbound data links, MultipleInputFeatureRequirement must be specified in the wf or wf step requirements.

If the sink parameter is an array, or named in a wf scatter operation, there may be multiple inbound data links listed in the source field. The values from the input links are merged depending on the method specified in the linkMerge field. If not specified, the default method is “merge_nested”.

- `merge_nested` The input must be an array consisting of exactly one entry for each input link. If “merge_nested” is specified with a single link, the value from the link must be wrapped in a single-item list.
- `merge_flattened` 1. The source and sink parameters must be compatible types, or the source type must be compatible with single element from the “items” type of the destination array parameter. 2. Source parameters which are arrays are concatenated. Source parameters which are single element types are appended as single elements.

id

A unique identifier for this wf input parameter.

source

Specifies one or more wf parameters that will provide input to the underlying step parameter.

link_merge

The method to use to merge multiple inbound links into a single array. If not specified, the default method is “merge_nested”.

default

The default value for this parameter to use if either there is no source field, or the value produced by the source is null. The default must be applied prior to scattering or evaluating valueFrom.

value_from

To use valueFrom, StepInputExpressionRequirement must be specified in the wf or wf step requirements.

If valueFrom is a constant string value, use this as the value for this input parameter.

If valueFrom is a parameter reference or expression, it must be evaluated to yield the actual value to be assigned to the input field.

The self value of in the parameter reference or expression must be the value of the parameter(s) specified in the source field, or null if there is no source field.

The value of inputs in the parameter reference or expression must be the input object to the wf step after assigning the source values, applying default, and then scattering. The order of evaluating valueFrom among step input parameters is undefined and the result of evaluating valueFrom on a parameter must not be visible to evaluation of valueFrom on other parameters.

```
class sbg.cwl.v1_0.wf.ExpressionTool(expression, cwl_version='v1.0', inputs=None,
                                     outputs=None, id=None, requirements=None,
                                     hints=None, label=None, doc=None, **kwargs)
```

Bases: sbg.cwl.v1_0.app.App

Execute an expression as a Workflow step.

```
class_ = 'ExpressionTool'
```

get_requirements (*value*)

expression

A long, human-readable description of this process object.

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

S

`sbg.cwl.sbg.session`, [34](#)
`sbg.cwl.v1_0`, [11](#)
`sbg.cwl.v1_0.cmd`, [40](#)
`sbg.cwl.v1_0.schema`, [36](#)
`sbg.cwl.v1_0.wf`, [44](#)

A

[add_connection\(\)](#) (sbg.cwl.v1_0.wf.Workflow method), [45](#)
[add_connection\(\)](#) (sbg.cwl.v1_0.Workflow method), [26](#)
[add_env_var\(\)](#) (sbg.cwl.v1_0.App method), [31](#)
[add_expression_lib\(\)](#) (sbg.cwl.v1_0.App method), [31](#)
[add_file\(\)](#) (sbg.cwl.v1_0.App method), [30](#)
[add_hints\(\)](#) (sbg.cwl.sbg.session.Session static method), [36](#)
[add_hints\(\)](#) (sbg.cwl.v1_0.App method), [30](#)
[add_in_workdir\(\)](#) (sbg.cwl.v1_0.App method), [32](#)
[add_input\(\)](#) (sbg.cwl.v1_0.App method), [32](#)
[add_input_json\(\)](#) (sbg.cwl.v1_0.App method), [30](#)
[add_locals\(\)](#) (sbg.cwl.v1_0.cmd.CommandLineTool method), [40](#)
[add_locals\(\)](#) (sbg.cwl.v1_0.CommandLineTool method), [16](#)
[add_output\(\)](#) (sbg.cwl.v1_0.App method), [32](#)
[add_requirement\(\)](#) (sbg.cwl.v1_0.App method), [32](#)
[add_requirement\(\)](#) (sbg.cwl.v1_0.wf.Workflow method), [45](#)
[add_requirement\(\)](#) (sbg.cwl.v1_0.Workflow method), [26](#)
[add_sbg_namespace\(\)](#) (sbg.cwl.v1_0.App static method), [31](#)
[add_step\(\)](#) (sbg.cwl.v1_0.wf.Workflow method), [45](#)
[add_step\(\)](#) (sbg.cwl.v1_0.Workflow method), [26](#)
[Any](#) (class in sbg.cwl.v1_0), [33](#)
[ANY](#) (sbg.cwl.v1_0.Primitive attribute), [12](#)
[App](#) (class in sbg.cwl.v1_0), [30](#)
[arguments](#) (sbg.cwl.v1_0.cmd.CommandLineTool attribute), [41](#)
[arguments](#) (sbg.cwl.v1_0.CommandLineTool attribute), [17](#)
[Array](#) (class in sbg.cwl.v1_0), [34](#)
[ArrayBase](#) (class in sbg.cwl.v1_0.schema), [39](#)

B

[base_command](#) (sbg.cwl.v1_0.cmd.CommandLineTool attribute), [41](#)

[base_command](#) (sbg.cwl.v1_0.CommandLineTool attribute), [17](#)

[Bool](#) (class in sbg.cwl.v1_0), [33](#)

[BOOLEAN](#) (sbg.cwl.v1_0.Primitive attribute), [12](#)

C

[calc_hash\(\)](#) (sbg.cwl.v1_0.Cwl method), [11](#)
[class_](#) (sbg.cwl.v1_0.cmd.CommandLineTool attribute), [40](#)
[class_](#) (sbg.cwl.v1_0.CommandLineTool attribute), [16](#)
[class_](#) (sbg.cwl.v1_0.Docker attribute), [21](#)
[class_](#) (sbg.cwl.v1_0.EnvVar attribute), [19](#)
[class_](#) (sbg.cwl.v1_0.ExpressionTool attribute), [27](#)
[class_](#) (sbg.cwl.v1_0.InitialWorkDir attribute), [20](#)
[class_](#) (sbg.cwl.v1_0.InlineJavascript attribute), [22](#)
[class_](#) (sbg.cwl.v1_0.MultipleInputFeature attribute), [27](#)
[class_](#) (sbg.cwl.v1_0.Resource attribute), [22](#)
[class_](#) (sbg.cwl.v1_0.ScatterFeature attribute), [27](#)
[class_](#) (sbg.cwl.v1_0.SchemaDef attribute), [19](#)
[class_](#) (sbg.cwl.v1_0.ShellCommand attribute), [23](#)
[class_](#) (sbg.cwl.v1_0.Software attribute), [19](#)
[class_](#) (sbg.cwl.v1_0.StepInputExpression attribute), [27](#)
[class_](#) (sbg.cwl.v1_0.SubworkflowFeature attribute), [27](#)
[class_](#) (sbg.cwl.v1_0.wf.ExpressionTool attribute), [48](#)
[class_](#) (sbg.cwl.v1_0.wf.MultipleInputFeature attribute), [46](#)
[class_](#) (sbg.cwl.v1_0.wf.ScatterFeature attribute), [46](#)
[class_](#) (sbg.cwl.v1_0.wf.StepInputExpression attribute), [46](#)
[class_](#) (sbg.cwl.v1_0.wf.SubworkflowFeature attribute), [46](#)
[class_](#) (sbg.cwl.v1_0.wf.Workflow attribute), [44](#)
[class_](#) (sbg.cwl.v1_0.Workflow attribute), [26](#)
[CommandInput](#) (class in sbg.cwl.v1_0), [15](#)
[CommandInput](#) (class in sbg.cwl.v1_0.cmd), [42](#)
[CommandLineTool](#) (class in sbg.cwl.v1_0), [16](#)
[CommandLineTool](#) (class in sbg.cwl.v1_0.cmd), [40](#)
[CommandOutput](#) (class in sbg.cwl.v1_0), [18](#)
[CommandOutput](#) (class in sbg.cwl.v1_0.cmd), [43](#)
[cores_max](#) (sbg.cwl.v1_0.Resource attribute), [22](#)

cores_min (sbg.cwl.v1_0.Resource attribute), 22
create_app() (sbg.cwl.sbg.session.Session method), 35
create_file() (sbg.cwl.v1_0.App method), 30
Cwl (class in sbg.cwl.v1_0), 11
cwl_version (sbg.cwl.v1_0.App attribute), 33

D

default (sbg.cwl.v1_0.cmd.CommandInput attribute), 43
default (sbg.cwl.v1_0.CommandInput attribute), 16
default (sbg.cwl.v1_0.StepInput attribute), 24
default (sbg.cwl.v1_0.wf.StepInput attribute), 48
Dir (class in sbg.cwl.v1_0), 34
DIRECTORY (sbg.cwl.v1_0.Primitive attribute), 12
Dirent (class in sbg.cwl.v1_0), 20
doc (sbg.cwl.v1_0.App attribute), 33
doc (sbg.cwl.v1_0.cmd.CommandInput attribute), 43
doc (sbg.cwl.v1_0.cmd.CommandOutput attribute), 44
doc (sbg.cwl.v1_0.CommandInput attribute), 16
doc (sbg.cwl.v1_0.CommandOutput attribute), 19
doc (sbg.cwl.v1_0.schema.RecordFieldBase attribute), 38
doc (sbg.cwl.v1_0.Step attribute), 25
doc (sbg.cwl.v1_0.wf.Step attribute), 47
Docker (class in sbg.cwl.v1_0), 21
docker_file (sbg.cwl.v1_0.Docker attribute), 21
docker_image_id (sbg.cwl.v1_0.Docker attribute), 22
docker_import (sbg.cwl.v1_0.Docker attribute), 22
docker_load (sbg.cwl.v1_0.Docker attribute), 21
docker_output_directory (sbg.cwl.v1_0.Docker attribute), 22
docker_pull (sbg.cwl.v1_0.Docker attribute), 21
DOTPRODUCT (sbg.cwl.v1_0.ScatterMethod attribute), 25
DOTPRODUCT (sbg.cwl.v1_0.wf.ScatterMethod attribute), 46
DOUBLE (sbg.cwl.v1_0.Primitive attribute), 12
draft() (sbg.cwl.sbg.session.Session method), 35
dump() (sbg.cwl.v1_0.Cwl method), 11

E

entry (sbg.cwl.v1_0.Dirent attribute), 20
entryname (sbg.cwl.v1_0.Dirent attribute), 21
Enum (class in sbg.cwl.v1_0), 34
EnumBase (class in sbg.cwl.v1_0.schema), 39
env_def (sbg.cwl.v1_0.EnvVar attribute), 19
env_name (sbg.cwl.v1_0.EnvironmentDef attribute), 19
env_value (sbg.cwl.v1_0.EnvironmentDef attribute), 19
EnvironmentDef (class in sbg.cwl.v1_0), 19
EnvVar (class in sbg.cwl.v1_0), 19
expression (sbg.cwl.v1_0.ExpressionTool attribute), 27
expression (sbg.cwl.v1_0.wf.ExpressionTool attribute), 49
expression_lib (sbg.cwl.v1_0.InlineJavascript attribute), 22
ExpressionTool (class in sbg.cwl.v1_0), 27

ExpressionTool (class in sbg.cwl.v1_0.wf), 48

F

fields (sbg.cwl.v1_0.schema.RecordBase attribute), 38
File (class in sbg.cwl.v1_0), 34
FILE (sbg.cwl.v1_0.Primitive attribute), 12
find_requirement() (sbg.cwl.v1_0.App method), 30
FLAT_CROSSPRODUCT (sbg.cwl.v1_0.ScatterMethod attribute), 25
FLAT_CROSSPRODUCT (sbg.cwl.v1_0.wf.ScatterMethod attribute), 46
Float (class in sbg.cwl.v1_0), 33
FLOAT (sbg.cwl.v1_0.Primitive attribute), 12
format (sbg.cwl.v1_0.cmd.CommandInput attribute), 43
format (sbg.cwl.v1_0.cmd.CommandOutput attribute), 44
format (sbg.cwl.v1_0.CommandInput attribute), 16
format (sbg.cwl.v1_0.CommandOutput attribute), 19
from_bash() (sbg.cwl.v1_0.cmd.CommandLineTool class method), 41
from_bash() (sbg.cwl.v1_0.CommandLineTool class method), 17

G

get_field_cls() (sbg.cwl.v1_0.InputRecord method), 29
get_field_cls() (sbg.cwl.v1_0.OutputRecord method), 29
get_field_cls() (sbg.cwl.v1_0.schema.InputRecord method), 38
get_field_cls() (sbg.cwl.v1_0.schema.OutputRecord method), 38
get_field_cls() (sbg.cwl.v1_0.schema.RecordBase method), 38
get_fields() (sbg.cwl.v1_0.schema.RecordBase method), 38
get_input() (sbg.cwl.v1_0.App method), 30
get_items() (sbg.cwl.v1_0.InputArray method), 29
get_items() (sbg.cwl.v1_0.OutputArray method), 29
get_items() (sbg.cwl.v1_0.schema.ArrayBase method), 39
get_items() (sbg.cwl.v1_0.schema.InputArray method), 39
get_items() (sbg.cwl.v1_0.schema.OutputArray method), 39
get_output() (sbg.cwl.v1_0.App method), 30
get_port() (sbg.cwl.v1_0.App method), 30
get_requirements() (sbg.cwl.v1_0.App method), 33
get_requirements() (sbg.cwl.v1_0.cmd.CommandLineTool method), 41
get_requirements() (sbg.cwl.v1_0.CommandLineTool method), 17
get_requirements() (sbg.cwl.v1_0.ExpressionTool method), 27
get_requirements() (sbg.cwl.v1_0.wf.ExpressionTool method), 48

- `get_requirements()` (sbg.cwl.v1_0.wf.Workflow method), 45
 - `get_requirements()` (sbg.cwl.v1_0.Workflow method), 26
 - `get_step()` (sbg.cwl.v1_0.wf.Workflow method), 45
 - `get_step()` (sbg.cwl.v1_0.Workflow method), 26
 - `get_type()` (sbg.cwl.v1_0.InputRecordField method), 28
 - `get_type()` (sbg.cwl.v1_0.OutputRecordField method), 29
 - `get_type()` (sbg.cwl.v1_0.schema.InputRecordField method), 39
 - `get_type()` (sbg.cwl.v1_0.schema.OutputRecordField method), 39
 - `get_type()` (sbg.cwl.v1_0.schema.RecordFieldBase method), 39
 - `get_unarchive_argument()` (sbg.cwl.v1_0.cmd.CommandLineTool static method), 40
 - `get_unarchive_argument()` (sbg.cwl.v1_0.CommandLineTool static method), 16
 - `glob` (sbg.cwl.v1_0.OutputBinding attribute), 29
 - `glob` (sbg.cwl.v1_0.schema.OutputBinding attribute), 38
- ## H
- `hints` (sbg.cwl.v1_0.App attribute), 33
 - `hints` (sbg.cwl.v1_0.Step attribute), 25
 - `hints` (sbg.cwl.v1_0.wf.Step attribute), 47
- ## I
- `id` (sbg.cwl.v1_0.App attribute), 33
 - `id` (sbg.cwl.v1_0.cmd.CommandInput attribute), 42
 - `id` (sbg.cwl.v1_0.cmd.CommandOutput attribute), 43
 - `id` (sbg.cwl.v1_0.CommandInput attribute), 15
 - `id` (sbg.cwl.v1_0.CommandOutput attribute), 18
 - `id` (sbg.cwl.v1_0.Step attribute), 25
 - `id` (sbg.cwl.v1_0.StepInput attribute), 24
 - `id` (sbg.cwl.v1_0.StepOutput attribute), 24
 - `id` (sbg.cwl.v1_0.wf.Step attribute), 47
 - `id` (sbg.cwl.v1_0.wf.StepInput attribute), 48
 - `id` (sbg.cwl.v1_0.wf.StepOutput attribute), 47
 - `in_` (sbg.cwl.v1_0.Step attribute), 25
 - `in_` (sbg.cwl.v1_0.wf.Step attribute), 47
 - `inherit_metadata()` (in module sbg.cwl.v1_0), 11
 - `inherit_metadata()` (sbg.cwl.v1_0.App method), 31
 - `init_api()` (sbg.cwl.sbg.session.Session static method), 35
 - `InitialWorkDir` (class in sbg.cwl.v1_0), 20
 - `InlineJavascript` (class in sbg.cwl.v1_0), 22
 - `input_binding` (sbg.cwl.v1_0.cmd.CommandInput attribute), 43
 - `input_binding` (sbg.cwl.v1_0.CommandInput attribute), 16
 - `input_binding` (sbg.cwl.v1_0.InputArray attribute), 29
 - `input_binding` (sbg.cwl.v1_0.InputEnum attribute), 29
 - `input_binding` (sbg.cwl.v1_0.InputRecordField attribute), 28
 - `input_binding` (sbg.cwl.v1_0.schema.InputArray attribute), 39
 - `input_binding` (sbg.cwl.v1_0.schema.InputEnum attribute), 40
 - `input_binding` (sbg.cwl.v1_0.schema.InputRecordField attribute), 39
 - `input_schema_item()` (in module sbg.cwl.v1_0.schema), 36
 - `InputArray` (class in sbg.cwl.v1_0), 29
 - `InputArray` (class in sbg.cwl.v1_0.schema), 39
 - `InputBinding` (class in sbg.cwl.v1_0), 27
 - `InputBinding` (class in sbg.cwl.v1_0.schema), 36
 - `InputEnum` (class in sbg.cwl.v1_0), 29
 - `InputEnum` (class in sbg.cwl.v1_0.schema), 40
 - `InputRecord` (class in sbg.cwl.v1_0), 28
 - `InputRecord` (class in sbg.cwl.v1_0.schema), 38
 - `InputRecordField` (class in sbg.cwl.v1_0), 28
 - `InputRecordField` (class in sbg.cwl.v1_0.schema), 39
 - `inputs` (sbg.cwl.v1_0.App attribute), 32
 - `Int` (class in sbg.cwl.v1_0), 33
 - `INT` (sbg.cwl.v1_0.Primitive attribute), 12
 - `is_number()` (in module sbg.cwl.v1_0), 12
 - `is_primitive()` (in module sbg.cwl.v1_0), 12
 - `is_required()` (sbg.cwl.v1_0.App static method), 31
 - `is_scattered()` (sbg.cwl.v1_0.Step method), 25
 - `is_scattered()` (sbg.cwl.v1_0.wf.Step method), 46
 - `item_separator` (sbg.cwl.v1_0.InputBinding attribute), 28
 - `item_separator` (sbg.cwl.v1_0.schema.InputBinding attribute), 37
 - `items_` (sbg.cwl.v1_0.schema.ArrayBase attribute), 39
- ## J
- `json_dump()` (sbg.cwl.v1_0.Cwl method), 11
- ## L
- `label` (sbg.cwl.v1_0.App attribute), 33
 - `label` (sbg.cwl.v1_0.cmd.CommandInput attribute), 42
 - `label` (sbg.cwl.v1_0.cmd.CommandOutput attribute), 43
 - `label` (sbg.cwl.v1_0.CommandInput attribute), 15
 - `label` (sbg.cwl.v1_0.CommandOutput attribute), 18
 - `label` (sbg.cwl.v1_0.schema.ArrayBase attribute), 39
 - `label` (sbg.cwl.v1_0.schema.EnumBase attribute), 39
 - `label` (sbg.cwl.v1_0.schema.RecordBase attribute), 38
 - `label` (sbg.cwl.v1_0.Step attribute), 25
 - `label` (sbg.cwl.v1_0.wf.Step attribute), 47
 - `link_merge` (sbg.cwl.v1_0.StepInput attribute), 24
 - `link_merge` (sbg.cwl.v1_0.wf.StepInput attribute), 48
 - `link_merge` (sbg.cwl.v1_0.wf.WorkflowOutput attribute), 45
 - `link_merge` (sbg.cwl.v1_0.WorkflowOutput attribute), 23
 - `link_merge()` (sbg.cwl.v1_0.Step method), 24
 - `link_merge()` (sbg.cwl.v1_0.wf.Step method), 46
 - `listing` (sbg.cwl.v1_0.InitialWorkDir attribute), 20
 - `load()` (in module sbg.cwl.v1_0), 11

load_contents (sbg.cwl.v1_0.InputBinding attribute), 28
load_contents (sbg.cwl.v1_0.OutputBinding attribute), 29
load_contents (sbg.cwl.v1_0.schema.InputBinding attribute), 37
load_contents (sbg.cwl.v1_0.schema.OutputBinding attribute), 38
LONG (sbg.cwl.v1_0.Primitive attribute), 12

M

MERGE_FLATTENED (sbg.cwl.v1_0.MergeMethod attribute), 23
MERGE_FLATTENED (sbg.cwl.v1_0.wf.MergeMethod attribute), 46
MERGE_NESTED (sbg.cwl.v1_0.MergeMethod attribute), 23
MERGE_NESTED (sbg.cwl.v1_0.wf.MergeMethod attribute), 46
MergeMethod (class in sbg.cwl.v1_0), 23
MergeMethod (class in sbg.cwl.v1_0.wf), 46
MultipleInputFeature (class in sbg.cwl.v1_0), 27
MultipleInputFeature (class in sbg.cwl.v1_0.wf), 46

N

name (sbg.cwl.v1_0.schema.RecordFieldBase attribute), 38
NESTED_CROSSPRODUCT (sbg.cwl.v1_0.ScatterMethod attribute), 25
NESTED_CROSSPRODUCT (sbg.cwl.v1_0.wf.ScatterMethod attribute), 46
NULL (sbg.cwl.v1_0.Primitive attribute), 12

O

out (sbg.cwl.v1_0.Step attribute), 25
out (sbg.cwl.v1_0.wf.Step attribute), 47
outdir_max (sbg.cwl.v1_0.Resource attribute), 23
outdir_min (sbg.cwl.v1_0.Resource attribute), 23
output_binding (sbg.cwl.v1_0.cmd.CommandOutput attribute), 44
output_binding (sbg.cwl.v1_0.CommandOutput attribute), 19
output_binding (sbg.cwl.v1_0.OutputArray attribute), 29
output_binding (sbg.cwl.v1_0.OutputEnum attribute), 29
output_binding (sbg.cwl.v1_0.OutputRecordField attribute), 29
output_binding (sbg.cwl.v1_0.schema.OutputArray attribute), 39
output_binding (sbg.cwl.v1_0.schema.OutputEnum attribute), 40
output_binding (sbg.cwl.v1_0.schema.OutputRecordField attribute), 39
output_eval (sbg.cwl.v1_0.OutputBinding attribute), 29

output_eval (sbg.cwl.v1_0.schema.OutputBinding attribute), 38
output_source (sbg.cwl.v1_0.wf.WorkflowOutput attribute), 45
output_source (sbg.cwl.v1_0.WorkflowOutput attribute), 23
OutputArray (class in sbg.cwl.v1_0), 29
OutputArray (class in sbg.cwl.v1_0.schema), 39
OutputBinding (class in sbg.cwl.v1_0), 29
OutputBinding (class in sbg.cwl.v1_0.schema), 37
OutputEnum (class in sbg.cwl.v1_0), 29
OutputEnum (class in sbg.cwl.v1_0.schema), 40
OutputRecord (class in sbg.cwl.v1_0), 29
OutputRecord (class in sbg.cwl.v1_0.schema), 38
OutputRecordField (class in sbg.cwl.v1_0), 29
OutputRecordField (class in sbg.cwl.v1_0.schema), 39
outputs (sbg.cwl.v1_0.App attribute), 33

P

package (sbg.cwl.v1_0.SoftwarePackage attribute), 19
packages (sbg.cwl.v1_0.Software attribute), 19
permanent_fail_codes (sbg.cwl.v1_0.cmd.CommandLineTool attribute), 42
permanent_fail_codes (sbg.cwl.v1_0.CommandLineTool attribute), 18
position (sbg.cwl.v1_0.InputBinding attribute), 28
position (sbg.cwl.v1_0.schema.InputBinding attribute), 37
prefix (sbg.cwl.v1_0.InputBinding attribute), 28
prefix (sbg.cwl.v1_0.schema.InputBinding attribute), 37
Primitive (class in sbg.cwl.v1_0), 12

R

ram_max (sbg.cwl.v1_0.Resource attribute), 22
ram_min (sbg.cwl.v1_0.Resource attribute), 22
Record (class in sbg.cwl.v1_0), 34
RecordBase (class in sbg.cwl.v1_0.schema), 38
RecordFieldBase (class in sbg.cwl.v1_0.schema), 38
requirements (sbg.cwl.v1_0.App attribute), 33
requirements (sbg.cwl.v1_0.Step attribute), 25
requirements (sbg.cwl.v1_0.wf.Step attribute), 47
resolve() (sbg.cwl.v1_0.Cwl method), 11
Resource (class in sbg.cwl.v1_0), 22
run (sbg.cwl.v1_0.Step attribute), 25
run (sbg.cwl.v1_0.wf.Step attribute), 47
run() (sbg.cwl.sbg.session.Session method), 36

S

sbg.cwl.sbg.session (module), 34
sbg.cwl.v1_0 (module), 11
sbg.cwl.v1_0.cmd (module), 40
sbg.cwl.v1_0.schema (module), 36
sbg.cwl.v1_0.wf (module), 44
scatter (sbg.cwl.v1_0.Step attribute), 25

- scatter (sbg.cwl.v1_0.wf.Step attribute), 47
 - scatter() (sbg.cwl.v1_0.wf.Workflow method), 45
 - scatter() (sbg.cwl.v1_0.Workflow method), 26
 - scatter_method (sbg.cwl.v1_0.Step attribute), 25
 - scatter_method (sbg.cwl.v1_0.wf.Step attribute), 47
 - ScatterFeature (class in sbg.cwl.v1_0), 27
 - ScatterFeature (class in sbg.cwl.v1_0.wf), 46
 - ScatterMethod (class in sbg.cwl.v1_0), 25
 - ScatterMethod (class in sbg.cwl.v1_0.wf), 46
 - SchemaBase (class in sbg.cwl.v1_0.schema), 38
 - SchemaDef (class in sbg.cwl.v1_0), 19
 - secondary_files (sbg.cwl.v1_0.cmd.CommandInput attribute), 42
 - secondary_files (sbg.cwl.v1_0.cmd.CommandOutput attribute), 43
 - secondary_files (sbg.cwl.v1_0.CommandInput attribute), 15
 - secondary_files (sbg.cwl.v1_0.CommandOutput attribute), 18
 - separate (sbg.cwl.v1_0.InputBinding attribute), 28
 - separate (sbg.cwl.v1_0.schema.InputBinding attribute), 37
 - Session (class in sbg.cwl.sbg.session), 34
 - set_required() (in module sbg.cwl.v1_0.schema), 40
 - set_required() (sbg.cwl.v1_0.App static method), 31
 - set_secondary_files() (sbg.cwl.v1_0.App method), 31
 - shell_quote (sbg.cwl.v1_0.InputBinding attribute), 28
 - shell_quote (sbg.cwl.v1_0.schema.InputBinding attribute), 37
 - ShellCommand (class in sbg.cwl.v1_0), 23
 - Software (class in sbg.cwl.v1_0), 19
 - SoftwarePackage (class in sbg.cwl.v1_0), 19
 - source (sbg.cwl.v1_0.StepInput attribute), 24
 - source (sbg.cwl.v1_0.wf.StepInput attribute), 48
 - specs (sbg.cwl.v1_0.SoftwarePackage attribute), 20
 - stage_input() (sbg.cwl.v1_0.App method), 31
 - stderr (sbg.cwl.v1_0.cmd.CommandLineTool attribute), 42
 - stderr (sbg.cwl.v1_0.CommandLineTool attribute), 18
 - stdin (sbg.cwl.v1_0.cmd.CommandLineTool attribute), 41
 - stdin (sbg.cwl.v1_0.CommandLineTool attribute), 17
 - stdout (sbg.cwl.v1_0.cmd.CommandLineTool attribute), 42
 - stdout (sbg.cwl.v1_0.CommandLineTool attribute), 17
 - Step (class in sbg.cwl.v1_0), 24
 - Step (class in sbg.cwl.v1_0.wf), 46
 - StepInput (class in sbg.cwl.v1_0), 23
 - StepInput (class in sbg.cwl.v1_0.wf), 47
 - StepInputExpression (class in sbg.cwl.v1_0), 27
 - StepInputExpression (class in sbg.cwl.v1_0.wf), 46
 - StepOutput (class in sbg.cwl.v1_0), 24
 - StepOutput (class in sbg.cwl.v1_0.wf), 47
 - steps (sbg.cwl.v1_0.wf.Workflow attribute), 45
 - steps (sbg.cwl.v1_0.Workflow attribute), 27
 - streamable (sbg.cwl.v1_0.cmd.CommandInput attribute), 43
 - streamable (sbg.cwl.v1_0.cmd.CommandOutput attribute), 44
 - streamable (sbg.cwl.v1_0.CommandInput attribute), 15
 - streamable (sbg.cwl.v1_0.CommandOutput attribute), 18
 - String (class in sbg.cwl.v1_0), 33
 - STRING (sbg.cwl.v1_0.Primitive attribute), 12
 - SubworkflowFeature (class in sbg.cwl.v1_0), 27
 - SubworkflowFeature (class in sbg.cwl.v1_0.wf), 46
 - success_codes (sbg.cwl.v1_0.cmd.CommandLineTool attribute), 42
 - success_codes (sbg.cwl.v1_0.CommandLineTool attribute), 18
 - symbols (sbg.cwl.v1_0.schema.EnumBase attribute), 39
- ## T
- temporary_fail_codes (sbg.cwl.v1_0.cmd.CommandLineTool attribute), 42
 - temporary_fail_codes (sbg.cwl.v1_0.CommandLineTool attribute), 18
 - tmpdir_max (sbg.cwl.v1_0.Resource attribute), 23
 - tmpdir_min (sbg.cwl.v1_0.Resource attribute), 22
 - to_dict() (sbg.cwl.v1_0.Cwl method), 11
 - to_input() (sbg.cwl.v1_0.schema.ArrayBase method), 39
 - to_input() (sbg.cwl.v1_0.schema.EnumBase method), 39
 - to_input() (sbg.cwl.v1_0.schema.RecordBase method), 38
 - to_input() (sbg.cwl.v1_0.schema.RecordFieldBase method), 39
 - to_json() (sbg.cwl.v1_0.Cwl method), 11
 - to_output() (sbg.cwl.v1_0.schema.ArrayBase method), 39
 - to_output() (sbg.cwl.v1_0.schema.EnumBase method), 40
 - to_output() (sbg.cwl.v1_0.schema.RecordBase method), 38
 - to_output() (sbg.cwl.v1_0.schema.RecordFieldBase method), 39
 - to_tool() (in module sbg.cwl.v1_0), 14
 - tool() (in module sbg.cwl.v1_0), 12
 - tool_from() (in module sbg.cwl.v1_0), 12
 - type (sbg.cwl.v1_0.Any attribute), 34
 - type (sbg.cwl.v1_0.Array attribute), 34
 - type (sbg.cwl.v1_0.Bool attribute), 33
 - type (sbg.cwl.v1_0.cmd.CommandInput attribute), 43
 - type (sbg.cwl.v1_0.cmd.CommandOutput attribute), 44
 - type (sbg.cwl.v1_0.CommandInput attribute), 16
 - type (sbg.cwl.v1_0.CommandOutput attribute), 19
 - type (sbg.cwl.v1_0.Dir attribute), 34
 - type (sbg.cwl.v1_0.Enum attribute), 34
 - type (sbg.cwl.v1_0.File attribute), 34
 - type (sbg.cwl.v1_0.Float attribute), 33

type (sbg.cwl.v1_0.Int attribute), 33
type (sbg.cwl.v1_0.Record attribute), 34
type (sbg.cwl.v1_0.schema.ArrayBase attribute), 39
type (sbg.cwl.v1_0.schema.EnumBase attribute), 39
type (sbg.cwl.v1_0.schema.RecordBase attribute), 38
type (sbg.cwl.v1_0.schema.RecordFieldBase attribute),
38
type (sbg.cwl.v1_0.String attribute), 33
type (sbg.cwl.v1_0.Union attribute), 34
types (sbg.cwl.v1_0.SchemaDef attribute), 19

U

unarchive_bundle() (sbg.cwl.v1_0.cmd.CommandLineTool
method), 40
unarchive_bundle() (sbg.cwl.v1_0.CommandLineTool
method), 16
Union (class in sbg.cwl.v1_0), 34
UnionBase (class in sbg.cwl.v1_0.schema), 38

V

value_from (sbg.cwl.v1_0.InputBinding attribute), 28
value_from (sbg.cwl.v1_0.schema.InputBinding at-
tribute), 37
value_from (sbg.cwl.v1_0.StepInput attribute), 24
value_from (sbg.cwl.v1_0.wf.StepInput attribute), 48
version (sbg.cwl.v1_0.SoftwarePackage attribute), 20

W

Workflow (class in sbg.cwl.v1_0), 25
Workflow (class in sbg.cwl.v1_0.wf), 44
workflow() (in module sbg.cwl.v1_0), 13
WorkflowInput (class in sbg.cwl.v1_0), 23
WorkflowInput (class in sbg.cwl.v1_0.wf), 46
WorkflowOutput (class in sbg.cwl.v1_0), 23
WorkflowOutput (class in sbg.cwl.v1_0.wf), 45
writable (sbg.cwl.v1_0.Dirent attribute), 21