

# Package ‘Rcwl’

March 30, 2021

**Title** An R interface to the Common Workflow Language

**Version** 1.6.0

**Description** The Common Workflow Language (CWL) is an open standard for development of data analysis workflows that is portable and scalable across different tools and working environments. Rcwl provides a simple way to wrap command line tools and build CWL data analysis pipelines programmatically within R. It increases the ease of usage, development, and maintenance of CWL pipelines.

**Depends** R (>= 3.6), yaml, methods, S4Vectors

**Imports** utils, stats, BiocParallel, batchtools, DiagrammeR, shiny, R.utils, codetools

**License** GPL-2 | file LICENSE

**Encoding** UTF-8

**LazyData** true

**Suggests** testthat, knitr, rmarkdown, BiocStyle

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**biocViews** Software, WorkflowStep, ImmunoOncology

**SystemRequirements** python (>= 2.7), cwltool (>= 1.0.2018)

**git\_url** <https://git.bioconductor.org/packages/Rcwl>

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** 54e6062

**git\_last\_commit\_date** 2020-10-27

**Date/Publication** 2021-03-29

**Author** Qiang Hu [aut, cre],  
Qian Liu [aut]

**Maintainer** Qiang Hu <[qiang.hu@roswellpark.org](mailto:qiang.hu@roswellpark.org)>

## R topics documented:

<code>+,cwlStepParam,stepParam-method</code> . . . . .	2
<code>cwlParam-class</code> . . . . .	3
<code>cwlShiny</code> . . . . .	4
<code>cwlStepParam-class</code> . . . . .	5

cwlVersion . . . . .	6
InputArrayParam-class . . . . .	7
InputParam-class . . . . .	8
InputParamList-class . . . . .	10
OutputArrayParam-class . . . . .	11
OutputParam-class . . . . .	11
OutputParamList-class . . . . .	13
plotCWL . . . . .	13
Rcwl . . . . .	14
readCWL . . . . .	15
requireDocker . . . . .	15
runCWL . . . . .	17
runCWLBatch . . . . .	18
runs . . . . .	19
short . . . . .	19
Step . . . . .	20
stepInParam-class . . . . .	20
stepInParamList-class . . . . .	21
stepParam-class . . . . .	22
stepParamList-class . . . . .	22
steps . . . . .	23
writeCWL . . . . .	24

## Index 25

---

+,cwlStepParam,stepParam-method  
*Pipeline*

---

### Description

To build a pipeline by connecting multiple ‘stepParam’ to a ‘cwlStepParam’ object.

### Usage

```
## S4 method for signature 'cwlStepParam,stepParam'
e1 + e2
```

### Arguments

e1                    A ‘cwlStepParam’ object.  
e2                    A ‘stepParam’ object.

### Value

A ‘cwlStepParam’ object.

### See Also

[cwlStepParam](#)

---

cwlParam-class	<i>Parameters for CWL</i>
----------------	---------------------------

---

## Description

The main CWL parameter class and constructor for command tools. More details: <https://www.commonwl.org/v1.0/CommandLineTool.html>

## Usage

```
cwlParam(
  cwlVersion = "v1.0",
  cwlClass = "CommandLineTool",
  baseCommand = character(),
  requirements = list(),
  hints = list(),
  arguments = list(),
  id = character(),
  label = character(),
  inputs = InputParamList(),
  outputs = OutputParamList(),
  stdout = character(),
  expression = character(),
  extensions = list(),
  intent = list()
)
```

## Arguments

cwlVersion	CWL version
cwlClass	"CommandLineTool"
baseCommand	Specifies the program or R function to execute
requirements	A list of Requirement lists that apply to either the runtime environment or the workflow engine.
hints	Any or a list for the workflow engine.
arguments	Command line bindings which are not directly associated with input parameters.
id	The unique identifier for this process object.
label	A short, human-readable label of this process object.
inputs	A object of 'InputParamList'.
outputs	A object of 'OutputParamList'.
stdout	Capture the command's standard output stream to a file written to the designated output directory.
expression	Javascripts for ExpressionTool class.
extensions	A list of extensions and metadata
intent	An identifier for the type of computational operation, of this Process.

## Details

<https://www.commonwl.org/v1.0/CommandLineTool.html>

**Value**

A 'cwlParam' class object.

**Examples**

```
input1 <- InputParam(id = "sth")
echo <- cwlParam(baseCommand = "echo", inputs = InputParamList(input1))
```

---

cwlShiny

*cwlShiny*

---

**Description**

Function to generate shiny app automatically for a 'cwlParam' object.

**Usage**

```
cwlShiny(cwl, inputList = list(), upload = FALSE, ...)
```

**Arguments**

cwl	A cwlParam object.
inputList	a list of choices for the inputs of cwl object. The name of the list must match the inputs of the cwl object.
upload	Whether to upload file. If FALSE, the upload field will be text input (file path) instead of file input.
...	More options for 'runCWL'.

**Value**

A shiny webapp.

**Examples**

```
input1 <- InputParam(id = "sth")
echo <- cwlParam(baseCommand = "echo", inputs = InputParamList(input1))
echoApp <- cwlShiny(echo)
```

---

cwlStepParam-class      *cwlStepParam*

---

### Description

A workflow steps parameter, which connect multiple command line steps into a workflow. More details: `stepInParameterList`.

### Usage

```
cwlStepParam(
  cwlVersion = "v1.0",
  cwlClass = "Workflow",
  requirements = list(),
  id = character(),
  label = character(),
  doc = list(),
  intent = list(),
  hints = list(),
  arguments = list(),
  extensions = list(),
  inputs = InputParamList(),
  outputs = OutputParamList(),
  steps = stepParamList()
)
```

### Arguments

<code>cwlVersion</code>	CWL version
<code>cwlClass</code>	"Workflow".
<code>requirements</code>	Requirements that apply to either the runtime environment or the workflow engine.
<code>id</code>	The unique identifier for this process object.
<code>label</code>	A short, human-readable label of this object.
<code>doc</code>	A documentation string for this object.
<code>intent</code>	An identifier for the type of computational operation, of this Process.
<code>hints</code>	Any or a list for the workflow engine.
<code>arguments</code>	Command line bindings which are not directly associated with input parameters.
<code>extensions</code>	A list of extensions and metadata.
<code>inputs</code>	A object of 'InputParamList'.
<code>outputs</code>	A object of 'OutputParamList'.
<code>steps</code>	A list of 'stepParamList'.

### Value

An object of class 'cwlStepParam'.

**Examples**

```

input1 <- InputParam(id = "sth")
echo1 <- cwlParam(baseCommand = "echo",
                 inputs = InputParamList(input1))
input2 <- InputParam(id = "sthout", type = "File")
echo2 <- cwlParam(baseCommand = "echo",
                 inputs = InputParamList(input2),
                 stdout = "out.txt")
i1 <- InputParam(id = "sth")
o1 <- OutputParam(id = "out", type = "File", outputSource = "echo2/output")
wf <- cwlStepParam(inputs = InputParamList(i1),
                  outputs = OutputParamList(o1))
s1 <- Step(id = "echo1", run = echo1, In = list(sth = "sth"))
s2 <- Step(id = "echo2", run = echo2, In = list(sthout = "echo1/output"))
wf <- wf + s1 + s2

```

---

cwlVersion

*cwlParam methods*


---

**Description**

- cwlParam methods
- cwlVersion CWL document version
- cwlClass
- cwlClass
- baseCommand
- baseCommand
- arguments
- arguments
- hints
- hints
- requirements
- requirements
- stdout of cwlParam
- stdout of cwlParam
- Extensions and metadata of cwlParam

**Usage**

```

cwlVersion(cwl)

cwlVersion(cwl) <- value

cwlClass(cwl)

cwlClass(cwl) <- value

```

```
baseCommand(cwl)
baseCommand(cwl) <- value
arguments(cwl, step = NULL)
arguments(cwl, step = NULL) <- value
hints(cwl)
hints(cwl) <- value
requirements(cwl, step = NULL)
requirements(cwl, step = NULL) <- value
stdOut(cwl)
stdOut(cwl) <- value
extensions(cwl)
extensions(cwl) <- value
```

### Arguments

cwl	A 'cwlParam' object.
value	To assign a list of 'requirements' value.
step	To specify a step ID when 'cwl' is a workflow.

### Value

cwlVersion: cwl version  
cwlClass: CWL Class  
baseCommand: CWL baseCommand  
arguments: CWL arguments  
hints: CWL hints  
requirements: CWL requirements  
stdOut: CWL stdout  
extensions: A list of extensions or metadata

---

InputArrayParam-class *InputArrayParam*

---

### Description

Parameters for array inputs. To specify an array parameter, the array definition is nested under the type field with 'type: array' and items defining the valid data types that may appear in the array. More details: <https://www.commonwl.org/v1.0/CommandLineTool.html#CommandInputArraySchema>

**Usage**

```

InputArrayParam(
  label = "",
  type = "array",
  items = character(),
  prefix = "",
  separate = TRUE,
  itemSeparator = character(),
  valueFrom = character()
)

```

**Arguments**

label	A short description for this object
type	Must be "array".
items	Defines the type of the array elements.
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.
itemSeparator	Join the array elements into a single string with separator.
valueFrom	String or Expression.

**Value**

An object of class 'InputArrayParam'.

**Examples**

```
InputArrayParam(items = "string", prefix="-B=", separate = FALSE)
```

---

InputParam-class	<i>Input parameters InputParam</i>
------------------	------------------------------------

---

**Description**

parameter for a command tool. More details: <https://www.commonwl.org/v1.0/CommandLineTool.html#CommandInput>

**Usage**

```

InputParam(
  id,
  label = "",
  type = "string",
  doc = character(),
  secondaryFiles = character(),
  streamable = logical(),
  format = character(),
  loadListing = character(),
)

```



```

    loadContents = logical(),
    position = 0L,
    prefix = "",
    separate = TRUE,
    itemSeparator = character(),
    valueFrom = character(),
    shellQuote = logical(),
    default = character(),
    value = character()
)

## S4 method for signature 'cwlParam'
x$name

## S4 replacement method for signature 'cwlParam'
x$name <- value

```

### Arguments

id	The unique identifier for this parameter object.
label	A short, human-readable label of this object.
type	valid types of data that may be assigned to this parameter.
doc	Optional. This argument takes an arbitrary documentation as a note for this object.
secondaryFiles	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.
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.
format	Only valid when type: File or is an array of items: File.
loadListing	Only valid when type: Directory or is an array of items: Directory.
loadContents	Only valid when type: File or is an array of items: File.
position	The position for this parameter.
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.
itemSeparator	Join the array elements into a single string with the elements separated by by itemSeparator.
valueFrom	String or Expression.
shellQuote	If ShellCommandRequirement is in the requirements for the current command, this controls whether the value is quoted on the command line (default is true).
default	The default value for this parameter
value	Assigned value for this parameter
x	A 'cwlParam' object.
name	One one of input list

**Value**

An object of class 'InputParam'.

**Examples**

```
input1 <- InputParam(id = "sth")
```

---

InputParamList-class    *InputParamList*

---

**Description**

InputParamList

InputParamList A list of InputParam

inputs

**Usage**

```
InputParamList(...)
```

```
inputs(cwl)
```

**Arguments**

...                    The InputParam objects.

cwl                    A cwlParam object

**Value**

An object of class 'InputParamList'.

inputs: A list of 'InputParam'.

**Examples**

```
input1 <- InputParam(id = "sth")
InputParamList(input1)
## Inputs
input1 <- InputParam(id = "sth")
echo <- cwlParam(baseCommand = "echo", inputs = InputParamList(input1))
inputs(echo)
```

---

 OutputArrayParam-class

*Output array parameters*


---

### Description

Parameters for array outputs. More details: <https://www.commonwl.org/v1.0/CommandLineTool.html#CommandOutput>

### Usage

```
OutputArrayParam(
  label = character(),
  type = "array",
  items = character(),
  glob = character(),
  loadContents = logical(),
  outputEval = character()
)
```

### Arguments

label	A short, human-readable label of this object.
type	Must be "array".
items	Defines the type of the array elements.
glob	Pattern to find files relative to the output directory.
loadContents	Read text from globbed file.
outputEval	Evaluate an expression to generate the output value.

### Value

An object of class 'OutputArrayParam'.

### Examples

```
b <- OutputParam(id = "b", type = OutputArrayParam(items = "File"), glob = "*.txt")
```

---

 OutputParam-class

*Output parameters*


---

### Description

An output parameter for a Command Line Tool. More details: <https://www.commonwl.org/v1.0/CommandLineTool.html>

**Usage**

```

OutputParam(
  id = "output",
  label = character(),
  doc = character(),
  type = "stdout",
  format = character(),
  secondaryFiles = character(),
  streamable = logical(),
  glob = character(),
  loadContents = logical(),
  outputEval = character(),
  outputSource = character()
)

```

**Arguments**

id	The unique identifier for this parameter object.
label	A short, human-readable label of this object.
doc	A documentation string for this object, or an array of strings which should be concatenated.
type	Specify valid types of data that may be assigned to this parameter.
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 File object.
secondaryFiles	Provides a pattern or expression specifying files or directories. Only valid when type: File or is an array of items: File.
streamable	A value of true indicates that the file is read or written sequentially without seeking. Only valid when type: File or is an array of items: File.
glob	Pattern to find files relative to the output directory.
loadContents	Read text from globbed file.
outputEval	Evaluate an expression to generate the output value.
outputSource	Specifies one or more workflow parameters that supply the value of to the output parameter.

**Value**

An object of class 'OutputParam'.

**Examples**

```
o1 <- OutputParam(id = "file", type = "File", glob = "*.txt")
```

---

OutputParamList-class *OutputParamList*

---

### Description

OutputParamList  
 OutputParamList #' A list of InputParam  
 outputs The outputs of a cwlParam object

### Usage

```
OutputParamList(out = OutputParam(), ...)
outputs(cwl)
```

### Arguments

out	The default stdout parameter.
...	The InputParam objects.
cwl	A cwlParam object

### Value

An object of class 'OutputParamList'.  
 outputs: A list of 'OutputParam'.

### Examples

```
o1 <- OutputParam(id = "file", type = "File", glob = "*.txt")
OutputParamList(o1)
input1 <- InputParam(id = "sth")
echo <- cwlParam(baseCommand = "echo", inputs = InputParamList(input1))
outputs(echo)
```

---

plotCWL *plotCWL*

---

### Description

Function to plot cwlStepParam object.

### Usage

```
plotCWL(cwl, output = "graph", layout = "tree", ...)
```

**Arguments**

<code>cwl</code>	A <code>cwlStepParam</code> object to plot
<code>output</code>	A string specifying the output type. An option inherits from <code>'render_graph'</code> and can also be "mermaid".
<code>layout</code>	Layout from <code>'render_graph'</code> .
<code>...</code>	other parameters from <code>'mermaid'</code> or <code>'render_graph'</code> function

**Value**

A workflow plot.

**Examples**

```
input1 <- InputParam(id = "sth")
echo1 <- cwlParam(baseCommand = "echo",
                 inputs = InputParamList(input1))
input2 <- InputParam(id = "sthout", type = "File")
echo2 <- cwlParam(baseCommand = "echo",
                 inputs = InputParamList(input2),
                 stdout = "out.txt")
i1 <- InputParam(id = "sth")
o1 <- OutputParam(id = "out", type = "File", outputSource = "echo2/output")
wf <- cwlStepParam(inputs = InputParamList(i1),
                 outputs = OutputParamList(o1))
s1 <- Step(id = "echo1", run = echo1, In = list(sth = "sth"))
s2 <- Step(id = "echo2", run = echo2, In = list(sthout = "echo1/output"))
wf <- wf + s1 + s2
plotCWL(wf)
```

---

Rcwl

*Rcwl*


---

**Description**

An R package to wrap command line tools and build pipelines with Common Workflow Language.

**See Also**

[cwlParam](#)

[runCWL](#)

---

readCWL	<i>Read CWL Function to read CWL command or workflow files.</i>
---------	---

---

**Description**

Read CWL Function to read CWL command or workflow files.

**Usage**

```
readCWL(cwlfile)
```

**Arguments**

cwlfile            The cwl file to read.

**Value**

A object of class 'cwlParam' or 'cwlStepParam'.

**Examples**

```
input1 <- InputParam(id = "sth")
echo <- cwlParam(baseCommand = "echo",
                 inputs = InputParamList(input1))
tf <- tempfile()
writeCWL(echo, tf)
readCWL(paste0(tf, ".cwl"))
```

---

requireDocker	<i>requireDocker</i>
---------------	----------------------

---

**Description**

requireDocker  
 requireJS  
 requireSoftware  
 InitialWorkDirRequirement  
 Dirent class  
 Create manifest for configure files  
 SubworkflowFeatureRequirement  
 ScatterFeatureRequirement  
 MultipleInputFeatureRequirement  
 StepInputExpressionRequirement

**Usage**

```

requireDocker(
  docker = NULL,
  Load = NULL,
  File = NULL,
  Import = NULL,
  ImageId = NULL,
  OutputDir = NULL
)

requireJS(expressionLib = list())

requireSoftware(packages = list())

requireInitialWorkDir(listing = list())

Dirent(entryname = character(), entry, writable = FALSE)

requireManifest(inputID, sep = "\\n")

requireSubworkflow()

requireScatter()

requireMultipleInput()

requireStepInputExpression()

```

**Arguments**

docker	The docker pull address.
Load	dockerLoad
File	dockerFile
Import	dockerImport
ImageId	dockerImageId
OutputDir	dockerOutputDirectory
expressionLib	optional code
packages	The list of software to be configured.
listing	The list of files or directories.
entryname	The name of the file or subdirectory to create in the output directory.
entry	character or expression.
writable	boolean.
inputID	The input ID from corresponding 'InputParam'.
sep	The separator of the input files in the manifest config.

**Details**

<https://www.commonwl.org/v1.0/CommandLineTool.html#Dirent>



**Value**

A DockerRequirement list  
 A InlineJavascriptRequirement list  
 A SoftwareRequirement list  
 A InitialWorkDirRequirement list  
 A SubworkflowFeatureRequirement list  
 A ScatterFeatureRequirement list  
 A MultipleInputFeatureRequirement list  
 A StepInputExpressionRequirement list

**Examples**

```
p1 <- InputParam(id = "ifiles", type = "File[]?", position = -1)
CAT = cwlParam(baseCommand = "cat",
  requirements = list(requireDocker("alpine"), requireManifest("ifiles"), requireJS()),
  arguments = list("ifiles"),
  inputs = InputParamList(p1))
```

runCWL

*run cwlParam***Description**

Execute a cwlParam object with assigned inputs.

**Usage**

```
runCWL(
  cwl,
  prefix = tempfile(),
  cwlRunner = "cwltool",
  cwlTemp = NULL,
  outdir = ".",
  cwlArgs = character(),
  stdout = TRUE,
  stderr = TRUE,
  showLog = FALSE,
  docker = TRUE,
  ...
)
```

**Arguments**

cwl	A 'cwlParam' or 'cwlStepParam' object.
prefix	The prefix of 'cwl' and 'yaml' file to write.
cwlRunner	The path to the 'cwltool' or 'cwl-runner'. If not exists, the cwltool package will be installed by 'reticulate'.
cwlTemp	Path to keep temporary files. If a directory path is given, the temporary files will be kept in the directory.

outdir	Output directory, default current directory.
cwlArgs	The arguments for 'cwltool' or 'cwl-runner'. For example, "--debug" can work with 'cwltool' to show debug information.
stdout	standard output from 'system2'.
stderr	standard error from 'system2'. By setting it to "", the detailed running logs will return directly.
showLog	Whether to show log details to standard out. i.e. stderr = "".
docker	Whether to use docker, or "singularity" if use Singularity runtime to run container.
...	The other options from 'writeCWL' and 'system2'.

**Value**

A list of outputs from tools and logs from cwltool.

**Examples**

```
input1 <- InputParam(id = "sth")
echo <- cwlParam(baseCommand = "echo",
                 inputs = InputParamList(input1))
echo$sth <- "Hello World!"
## res <- runCWL(echo)
```

---

runCWLBatch

*run CWL with batchtools*


---

**Description**

run CWL with batchtools

**Usage**

```
runCWLBatch(
  cwl,
  outdir = getwd(),
  inputList,
  paramList = list(),
  BPPARAM = BatchtoolsParam(workers = lengths(inputList)[1]),
  ...
)
```

**Arguments**

cwl	A 'cwlParam' or 'cwlStepParam' object.
outdir	Directory to output results
inputList	An input list to run in parallel. The list names must be in the inputs of cwl. Jobs will be submitted in parallel for each element in the list. The output directory of each job will be made using the name of each element under the 'outdir'.
paramList	A parameter list for the cwl. The list names must be in the inputs of cwl.
BPPARAM	The options for 'BiocParallelParam'.
...	The options from runCWL.

**Value**

Results from computing nodes and logs from cwltool.

---

runs

*runs*

---

**Description**

The function to access all runs of a cwlStepParam object

**Usage**

runs(object)

**Arguments**

object            A cwlStepParam object.

**Value**

cwlParam objects or paths of CWL file.

**Examples**

```
s1 <- cwlStepParam()
runs(s1)
```

---

short

*short*

---

**Description**

The function to show short summary of cwlParam or cwlStepParam

**Usage**

short(object)

**Arguments**

object            An cwlParam or cwlStepParam object

**Value**

A short summary of an object of cwlParam or cwlStepParam.

**Examples**

```
s1 <- cwlStepParam()
short(s1)
```

---

Step	<i>Step function</i>
------	----------------------

---

**Description**

Function to assign value to 'stepParam' object.

**Usage**

```
Step(
  id,
  run = cwlParam(),
  In = list(),
  scatter = character(),
  scatterMethod = character()
)
```

**Arguments**

id	The id of 'stepParam' object.
run	A 'cwlParam' object for command tool, or path to a CWL file.
In	one or two layers of list.
scatter	character or a list. The inputs to be scattered.
scatterMethod	required if scatter is an array of more than one element. It can be one of "dot-product", "nested_crossproduct" and "flat_crossproduct". Details: <a href="https://www.commonwl.org/v1.0/">https://www.commonwl.org/v1.0/</a>

**Value**

An object of 'stepParam'.

**See Also**

[cwlStepParam](#)

---

stepInParam-class	<i>stepInParam</i>
-------------------	--------------------

---

**Description**

The input parameter of a workflow step. More details: <https://www.commonwl.org/v1.0/Workflow.html#WorkflowStepIn>

**Usage**

```
stepInParam(
  id,
  source = character(),
  linkMerge = character(),
  default = character(),
  valueFrom = character()
)
```

**Arguments**

id	A unique identifier for this workflow input parameter.
source	Specifies one or more workflow parameters that will provide input to the underlying step parameter.
linkMerge	The method to use to merge multiple inbound links into a single array.
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.
valueFrom	value from string or expression.

**Value**

An object of class 'stepInParam'.

**Examples**

```
s1 <- stepInParam(id = "s1")
```

---

stepInParamList-class *stepInParamList*

---

**Description**

stepInParamList

stepInParamList

**Usage**

```
stepInParamList(...)
```

**Arguments**

... A list of 'stepInParam' objects.

**Value**

An object of class 'stepInParamList'.

**Examples**

```
s1 <- stepInParam(id = "s1")
stepInParamList(s1)
```

---

stepParam-class	<i>stepParam</i>
-----------------	------------------

---

**Description**

A workflow step parameters. More details: <https://www.commonwl.org/v1.0/Workflow.html#WorkflowStep>

**Usage**

```
stepParam(
  id,
  run = cwlParam(),
  In = stepInParameterList(),
  Out = list(),
  scatter = character(),
  scatterMethod = character()
)
```

**Arguments**

id	The unique identifier for this workflow step.
run	A 'cwlParam' object or the path of a cwl file.
In	A 'stepInParameterList'.
Out	A list of outputs
scatter	character or a list. The inputs to be scattered.
scatterMethod	required if scatter is an array of more than one element. It can be one of "dot-product", "nested_crossproduct" and "flat_crossproduct". Details: <a href="https://www.commonwl.org/v1.0/">https://www.commonwl.org/v1.0/</a>

**Value**

An object of class 'stepParam'.

**Examples**

```
s1 <- stepParam(id = "s1")
```

---

stepParamList-class	<i>stepParamList</i>
---------------------	----------------------

---

**Description**

```
stepParamList
stepParamList
```

**Usage**

```
stepParamList(...)
```

**Arguments**

... A list of 'stepParam'.

**Value**

An object of class 'stepParamList'.

**Examples**

```
s1 <- stepParam(id = "s1")
stepParamList(s1)
```

---

steps	<i>Steps</i>
-------	--------------

---

**Description**

Function to extract step slots

**Usage**

```
steps(cwl)
```

```
steps(cwl) <- value
```

**Arguments**

cwl A cwlStepParam object.

value A list of steps.

**Value**

steps: A list of stepParam objects.

**See Also**

[cwlStepParam](#)

---

`writeCWL`*Write CWL*

---

**Description**

write 'cwlParam' to cwl and yml.

**Usage**

```
writeCWL(cwl, prefix, docker = TRUE, ...)
```

**Arguments**

<code>cwl</code>	A 'cwlParam' or 'cwlStepParam' object.
<code>prefix</code>	The prefix of 'cwl' and 'yaml' file to write.
<code>docker</code>	Whether to use docker.
<code>...</code>	Other options from 'yaml::write_yaml'.

**Value**

A CWL file and A YAML file.

**Examples**

```
input1 <- InputParam(id = "sth")
echo <- cwlParam(baseCommand = "echo",
                 inputs = InputParamList(input1))
tf <- tempfile()
writeCWL(echo, tf)
```



# Index

- [+, cwlStepParam, stepParam-method, 2](#)
  - [\\$, cwlParam-method \(InputParam-class\), 8](#)
  - [\\$<- , cwlParam-method \(InputParam-class\), 8](#)
- [arguments \(cwlVersion\), 6](#)
- [arguments<- \(cwlVersion\), 6](#)
- [baseCommand \(cwlVersion\), 6](#)
- [baseCommand<- \(cwlVersion\), 6](#)
- [cwlClass \(cwlVersion\), 6](#)
- [cwlClass<- \(cwlVersion\), 6](#)
- [cwlParam, 14](#)
- [cwlParam \(cwlParam-class\), 3](#)
- [cwlParam-class, 3](#)
- [cwlShiny, 4](#)
- [cwlStepParam, 2, 20, 23](#)
- [cwlStepParam \(cwlStepParam-class\), 5](#)
- [cwlStepParam-class, 5](#)
- [cwlVersion, 6](#)
- [cwlVersion<- \(cwlVersion\), 6](#)
- [Dirent \(requireDocker\), 15](#)
- [extensions \(cwlVersion\), 6](#)
- [extensions<- \(cwlVersion\), 6](#)
- [hints \(cwlVersion\), 6](#)
- [hints<- \(cwlVersion\), 6](#)
- [InputArrayParam \(InputArrayParam-class\), 7](#)
- [InputArrayParam-class, 7](#)
- [InputParam \(InputParam-class\), 8](#)
- [InputParam-class, 8](#)
- [InputParamList \(InputParamList-class\), 10](#)
- [InputParamList-class, 10](#)
- [inputs \(InputParamList-class\), 10](#)
- [OutputArrayParam \(OutputArrayParam-class\), 11](#)
- [OutputArrayParam-class, 11](#)
- [OutputParam \(OutputParam-class\), 11](#)
- [OutputParam-class, 11](#)
- [OutputParamList \(OutputParamList-class\), 13](#)
- [OutputParamList-class, 13](#)
- [outputs \(OutputParamList-class\), 13](#)
- [plotCWL, 13](#)
- [Rcwl, 14](#)
- [readCWL, 15](#)
- [requireDocker, 15](#)
- [requireInitialWorkDir \(requireDocker\), 15](#)
- [requireJS \(requireDocker\), 15](#)
- [requireManifest \(requireDocker\), 15](#)
- [requirements \(cwlVersion\), 6](#)
- [requirements<- \(cwlVersion\), 6](#)
- [requireMultipleInput \(requireDocker\), 15](#)
- [requireScatter \(requireDocker\), 15](#)
- [requireSoftware \(requireDocker\), 15](#)
- [requireStepInputExpression \(requireDocker\), 15](#)
- [requireSubworkflow \(requireDocker\), 15](#)
- [runCWL, 14, 17](#)
- [runCWLBatch, 18](#)
- [runs, 19](#)
- [short, 19](#)
- [stdout \(cwlVersion\), 6](#)
- [stdout<- \(cwlVersion\), 6](#)
- [Step, 20](#)
- [stepInParam \(stepInParam-class\), 20](#)
- [stepInParam-class, 20](#)
- [stepInParamList \(stepInParamList-class\), 21](#)
- [stepInParamList-class, 21](#)
- [stepParam \(stepParam-class\), 22](#)
- [stepParam-class, 22](#)
- [stepParamList \(stepParamList-class\), 22](#)
- [stepParamList-class, 22](#)
- [steps, 23](#)
- [steps<- \(steps\), 23](#)
- [writeCWL, 24](#)