
stpipe

Release 0.5.2.dev44+gd20b642.d20240321

STScI <help@stsci.edu>

Mar 21, 2024

CONTENTS:

1	API	1
1.1	stpipe API	1
2	Indices and tables	13
	Python Module Index	15
	Index	17

1.1 stpipe API

1.1.1 stpipe Package

Classes

<i>Pipeline</i> (*args, **kwargs)	A Pipeline is a way of combining a number of steps together.
<i>Step</i> ([name, parent, config_file, _validate_kwds])	

Pipeline

class `stpiper.Pipeline(*args, **kwargs)`

Bases: *Step*

A Pipeline is a way of combining a number of steps together.

See *Step.__init__* for the parameters.

Attributes Summary

<i>reference_file_types</i>	Collect the list of all reftypes for child Steps that are not skipped.
<i>spec</i>	
<i>step_defs</i>	

Methods Summary

<code>get_config_from_reference(dataset[, ...])</code>	Retrieve step parameters from reference database
<code>get_pars([full_spec])</code>	Retrieve the configuration parameters of a pipeline
<code>get_ref_override(reference_file_type)</code>	Return any override for <i>reference_file_type</i> for any of the steps in Pipeline <i>self</i> .
<code>load_spec_file([preserve_comments])</code>	
<code>merge_config(config, config_file)</code>	
<code>merge_pipeline_config(refcfg, ref_file)</code>	Merge the config parameters from a pipeline config reference file into the config obtained from each step
<code>set_input_filename(path)</code>	

Attributes Documentation

reference_file_types: `ClassVar`

Collect the list of all reftypes for child Steps that are not skipped. Overridden reftypes are included but handled normally later by the Pipeline version of the `get_ref_override()` method defined below.

spec = `'\n '`

step_defs: `ClassVar = {}`

Methods Documentation

classmethod `get_config_from_reference(dataset, disable=None, crds_observatory=None)`

Retrieve step parameters from reference database

Parameters

cls

[*jwst.stpipe.step.Step*] Either a class or instance of a class derived from *Step*.

dataset

[*jwst.datamodels.ModelBase*] A model of the input file. Metadata on this input file will be used by the CRDS “bestref” algorithm to obtain a reference file.

disable: `bool or None`

Do not retrieve parameters from CRDS. If `None`, check global settings.

crds_observatory

[`str`] Observatory name (`‘jwst’` or `‘roman’`).

Returns

step_parameters

[`configobj`] The parameters as retrieved from CRDS. If there is an issue, log as such and return an empty config obj.

get_pars(*full_spec=True*)

Retrieve the configuration parameters of a pipeline

Parameters are retrieved for the pipeline and all of its component steps.

Parameters**full_spec**

[bool] Return all parameters, including parent-specified parameters. If *False*, return only parameters specific to the pipeline and steps.

Returns**pars**

[dict] Keys are the parameters and values are the values.

get_ref_override(*reference_file_type*)

Return any override for *reference_file_type* for any of the steps in Pipeline *self*. OVERRIDES Step.

Returns

override_filepath or None.

classmethod load_spec_file(*preserve_comments=<stpipe.utilities._NotSet object>*)**classmethod merge_config**(*config, config_file*)**classmethod merge_pipeline_config**(*refcfg, ref_file*)

Merge the config parameters from a pipeline config reference file into the config obtained from each step

Parameters**cls**

[jwst.stpipe.pipeline.Pipeline class] The pipeline class

refcfg

[ConfigObj object] The ConfigObj created from crds cfg files from each of the steps in the pipeline

ref_file

[string] The name of the pipeline crds step config file

Returns

ConfigObj of the merged parameters, with those from the pipeline cfg having precedence over those from the individual steps

set_input_filename(*path*)

Step

class stpipe.Step(*name=None, parent=None, config_file=None, _validate_kwds=True, **kws*)

Bases: object

Create a *Step* instance.

Parameters**name**

[str, optional] The name of the Step instance. Used in logging messages and in cache file-names. If not provided, one will be generated based on the class name.

parent

[Step instance, optional] The parent step of this step. Used to determine a fully-qualified name for this step, and to determine the mode in which to run this step.

config_file

[str or pathlib.Path, optional] The path to the config file that this step was initialized with. Use to determine relative path names of other config files.

****kws**

[dict] Additional parameters to set. These will be set as member variables on the new Step instance.

Attributes Summary

<i>class_alias</i>	
<i>correction_pars</i>	
<i>input_dir</i>	
<i>log_records</i>	Retrieve logs from the most recent run of this step.
<i>make_output_path</i>	Return function that creates the output path
<i>name_format</i>	
<i>prefetch_references</i>	
<i>reference_file_types</i>	
<i>spec</i>	
<i>use_correction_pars</i>	

Methods Summary

<i>__call__</i> (*args)	Run handles the generic setup and teardown that happens with the running of each step.
<i>build_config</i> (input, **kwargs)	Build the ConfigObj to initialize a Step
<i>call</i> (*args, **kwargs)	Creates and runs a new instance of the class.
<i>closeout</i> ([to_close, to_del])	Close out step processing
<i>default_output_file</i> ([input_file])	Create a default filename based on the input name
<i>default_suffix</i> ()	Return a default suffix based on the step
<i>export_config</i> (filename[, include_metadata])	Export this step's parameters to an ASDF config file.
<i>finalize_result</i> (result, reference_files_used)	Hook that allows subclasses to set mission-specific metadata on each step result before that result is saved.
<i>from_cmdline</i> (args)	Create a step from a configuration file.
<i>from_config_file</i> (config_file[, parent, name])	Create a step from a configuration file.
<i>from_config_section</i> (config[, parent, name, ...])	Create a step from a configuration file fragment.
<i>get_config_from_reference</i> (dataset[, ...])	Retrieve step parameters from reference database
<i>get_config_reftype</i> ()	Get the CRDS reftype for this step's config reference.
<i>get_pars</i> ([full_spec])	Retrieve the configuration parameters of a step
<i>get_ref_override</i> (reference_file_type)	Determine and return any override for <i>reference_file_type</i> .

continues on next page

Table 1 – continued from previous page

<i>get_reference_file</i> (input_file, ...)	Get a reference file from CRDS.
<i>load_spec_file</i> ([preserve_comments])	
<i>make_input_path</i> (file_path)	Create an input path for a given file path
<i>merge_config</i> (config, config_file)	
<i>open_model</i> (init, **kwargs)	Open a datamodel
<i>prefetch</i> (*args)	Prefetch reference files, nominally called when self.prefetch_references is True.
<i>print_configspec</i> ()	
<i>process</i> (*args)	This is where real work happens.
<i>reference_uri_to_cache_path</i> (reference_uri, ...)	Convert an abstract CRDS reference URI to an absolute file path in the CRDS cache.
<i>remove_suffix</i> (name)	Remove a known Step filename suffix from a filename (if present).
<i>resolve_file_name</i> (file_name)	Resolve a file name expressed relative to this Step's configuration file.
<i>run</i> (*args)	Run handles the generic setup and teardown that happens with the running of each step.
<i>save_model</i> (model[, suffix, idx, ...])	Saves the given model using the step/pipeline's naming scheme
<i>search_attr</i> (attribute[, default, parent_first])	Return first non-None attribute in step hierarchy
<i>set_primary_input</i> (obj[, exclusive])	Sets the name of the master input file and input directory.
<i>update_pars</i> (parameters)	Update step parameters

Attributes Documentation

class_alias = None

correction_pars = None

input_dir

log_records

Retrieve logs from the most recent run of this step.

Returns

list of logging.LogRecord

make_output_path

Return function that creates the output path

name_format = None

prefetch_references = True

reference_file_types: ClassVar = []

```
spec = '\n pre_hooks = list(default=list()) # List of Step classes to run before
step\n post_hooks = list(default=list()) # List of Step classes to run after step\n
output_file = output_file(default=None) # File to save output to.\n output_dir =
string(default=None) # Directory path for output files\n output_ext = string() #
Default type of output\n output_use_model = boolean(default=False) # When saving use
`DataModel.meta.filename`\n output_use_index = boolean(default=True) # Append
index.\n save_results = boolean(default=False) # Force save results\n skip =
boolean(default=False) # Skip this step\n suffix = string(default=None) # Default
suffix for output files\n search_output_file = boolean(default=True) # Use
outputfile define in parent step\n input_dir = string(default=None) # Input
directory\n '
```

use_correction_pars = False

Methods Documentation

`__call__(*args)`

Run handles the generic setup and teardown that happens with the running of each step. The real work that is unique to each step type is done in the *process* method.

`classmethod build_config(input, **kwargs)`

Build the ConfigObj to initialize a Step

A Step config is built in the following order:

- CRDS parameter reference file
- Local parameter reference file
- Step keyword arguments

Parameters

input

[str or None] Input file

kwargs

[dict] Keyword arguments that specify Step parameters.

Returns

config, config_file

[ConfigObj, str] The configuration and the config filename.

`classmethod call(*args, **kwargs)`

Creates and runs a new instance of the class.

Gets a config file from CRDS if one is available

To set configuration parameters, pass a *config_file* path or keyword arguments. Keyword arguments override those in the specified *config_file*.

Any positional **args* will be passed along to the step's *process* method.

Note: this method creates a new instance of *Step* with the given *config_file* if supplied, plus any extra **args* and ***kwargs*. If you create an instance of a Step, set parameters, and then use this *call()* method, it will ignore previously-set parameters, as it creates a new instance of the class with only the *config_file*, **args* and ***kwargs* passed to the *call()* method.

If not used with a *config_file* or specific **args* and ***kwargs*, it would be better to use the *run* method, which does not create a new instance but simply runs the existing instance of the *Step* class.

closeout(*to_close=None, to_del=None*)

Close out step processing

Parameters

to_close

[[object(, ...)]] List of objects with a *close* method to execute The objects will also be deleted

to_del

[[object(, ...)]] List of objects to simply delete

Notes

Other operations, such as forced garbage collection will also be done.

default_output_file(*input_file=None*)

Create a default filename based on the input name

default_suffix()

Return a default suffix based on the step

export_config(*filename, include_metadata=False*)

Export this step's parameters to an ASDF config file.

Parameters

filename

[str or pathlib.Path] Path to config file.

include_metadata

[bool, optional] Set to True to include metadata that is required for submission to CRDS.

finalize_result(*result, reference_files_used*)

Hook that allows subclasses to set mission-specific metadata on each step result before that result is saved.

Parameters

result

[a datamodel that is an instance of AbstractDataModel or] collections.abc.Sequence One step result (potentially of many).

reference_files_used

[list of tuple] List of reference files used when running the step, each a tuple in the form (str reference type, str reference URI).

static from_cmdline(*args*)

Create a step from a configuration file.

Parameters

args

[list of str] Commandline arguments

Returns

step

[Step instance] If the config file has a *class* parameter, the return value will be as instance of that class.

Any parameters found in the config file will be set as member variables on the returned *Step* instance.

classmethod from_config_file(*config_file*, *parent=None*, *name=None*)

Create a step from a configuration file.

Parameters

config_file

[path or readable file-like object] The config file to load parameters from

parent

[Step instance, optional] The parent step of this step. Used to determine a fully-qualified name for this step, and to determine the mode in which to run this step.

name

[str, optional] If provided, use that name for the returned instance. If not provided, the following are tried (in order): - The *name* parameter in the config file - The filename of the config file - The name of returned class

Returns

step

[Step instance] If the config file has a *class* parameter, the return value will be as instance of that class. The *class* parameter in the config file must specify a subclass of *cls*. If the configuration file has no *class* parameter, then an instance of *cls* is returned.

Any parameters found in the config file will be set as member variables on the returned *Step* instance.

classmethod from_config_section(*config*, *parent=None*, *name=None*, *config_file=None*)

Create a step from a configuration file fragment.

Parameters

config

[configobj.Section instance] The config file fragment containing parameters for this step only.

parent

[Step instance, optional] The parent step of this step. Used to determine a fully-qualified name for this step, and to determine the mode in which to run this step.

name

[str, optional] If provided, use that name for the returned instance. If not provided, try the following (in order): - The *name* parameter in the config file fragment - The name of returned class

config_file

[str or pathlib.Path, optional] The path to the config file that created this step, if any. This is used to resolve relative file name parameters in the config file.

Returns

step

[instance of *cls*] Any parameters found in the config file fragment will be set as member variables on the returned *Step* instance.

classmethod `get_config_from_reference(dataset, disable=None, crds_observatory=None)`

Retrieve step parameters from reference database

Parameters

cls

[stpipe.Step] Either a class or instance of a class derived from *Step*.

dataset

[A datamodel that is an instance of AbstractDataModel] A model of the input file. Metadata on this input file will be used by the CRDS “bestref” algorithm to obtain a reference file.

disable: bool or None

Do not retrieve parameters from CRDS. If None, check global settings.

crds_observatory

[str] Observatory name (‘jwst’ or ‘roman’).

Returns

step_parameters

[configobj] The parameters as retrieved from CRDS. If there is an issue, log as such and return an empty config obj.

classmethod `get_config_reftype()`

Get the CRDS reftype for this step’s config reference.

Returns

str

get_pars(*full_spec=True*)

Retrieve the configuration parameters of a step

Parameters

full_spec

[bool] Return all parameters, including parent-specified parameters. If *False*, return only parameters specific to the step.

Returns

dict

Keys are the parameters and values are the values.

get_ref_override(*reference_file_type*)

Determine and return any override for *reference_file_type*.

Returns

override_filepath or None.

get_reference_file(*input_file, reference_file_type*)

Get a reference file from CRDS.

If the configuration file or commandline parameters override the reference file, it will be automatically used when calling this function.

Parameters

input_file

[a datamodel that is an instance of AbstractDataModel] A model of the input file. Metadata on this input file will be used by the CRDS “bestref” algorithm to obtain a reference file.

reference_file_type

[string] The type of reference file to retrieve. For example, to retrieve a flat field reference file, this would be 'flat'.

Returns**reference_file**

[path of reference file, a string]

classmethod **load_spec_file**(*preserve_comments=<stpipe.utilities._NotSet object>*)

make_input_path(*file_path*)

Create an input path for a given file path

If *file_path* has no directory path, use *self.input_dir* as the directory path.

Parameters**file_path**

[str or obj] The supplied file path to check and modify. If anything other than *str*, the object is simply passed back.

Returns**full_path**

[str or obj] File path using *input_dir* if the input had no directory path.

classmethod **merge_config**(*config, config_file*)

open_model(*init, **kwargs*)

Open a datamodel

Primarily a wrapper around *DataModel.open* to handle *Step* peculiarities

Parameters**init**

[object] The object to open

Returns**datamodel**

[instance of AbstractDataModel] Object opened as a datamodel

prefetch(*args)

Prefetch reference files, nominally called when *self.prefetch_references* is True. Can be called explicitly when *self.prefetch_references* is False.

classmethod **print_configspec**()

process(*args)

This is where real work happens. Every *Step* subclass has to override this method. The default behaviour is to raise a *NotImplementedError* exception.

classmethod **reference_uri_to_cache_path**(*reference_uri, observatory*)

Convert an abstract CRDS reference URI to an absolute file path in the CRDS cache. Reference URI's are typically output to dataset headers to record the reference files used.

e.g. 'crds://jwst_miri_flat_0177.fits' ->

'/grp/crds/cache/references/jwst/jwst_miri_flat_0177.fits'

The CRDS cache is typically located relative to env var *CRDS_PATH* with default value */grp/crds/cache*. See also <https://jwst-crds.stsci.edu>

static remove_suffix(*name*)

Remove a known Step filename suffix from a filename (if present).

Parameters

name
[str] Filename.

Returns

str
Filename with any known suffix removed.

str
Separator that delimited the original suffix.

resolve_file_name(*file_name*)

Resolve a file name expressed relative to this Step's configuration file.

run(*args)

Run handles the generic setup and teardown that happens with the running of each step. The real work that is unique to each step type is done in the *process* method.

save_model(*model*, *suffix=None*, *idx=None*, *output_file=None*, *force=False*, *format=None*, ***components*)

Saves the given model using the step/pipeline's naming scheme

Parameters

model
[a instance of AbstractDataModel] The model to save.

suffix
[str] The suffix to add to the filename.

idx
[object] Index identifier.

output_file
[str] Use this file name instead of what the Step default would be.

force
[bool] Regardless of whether *save_results* is *False* and no *output_file* is specified, try saving.

format
[str] The format of the file name. This is a format string that defines where *suffix* and the other components go in the file name. If *False*, it will be presumed *output_file* will have all the necessary formatting.

components
[dict] Other components to add to the file name.

Returns

output_paths
[[str, ...]] List of output file paths the model(s) were saved in.

search_attr(*attribute*, *default=None*, *parent_first=False*)

Return first non-None attribute in step hierarchy

Parameters

attribute

[str] The attribute to retrieve

default

[obj] If attribute is not found, the value to use

parent_first

[bool] If *True*, allow parent definition to override step version

Returns**value**

[obj] Attribute value or *default* if not found

set_primary_input(*obj*, *exclusive=True*)

Sets the name of the master input file and input directory. Used to generate output file names.

Parameters**obj**

[str, pathlib.Path, or instance of AbstractDataModel] The object to base the name on. If a *datamodel*, use *Datamodel.meta.filename*.

exclusive

[bool] If *True*, only set if an input name is not already used by a parent Step. Otherwise, always set.

update_pars(*parameters*)

Update step parameters

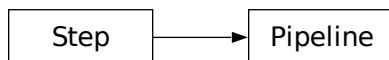
Only existing parameters are updated. Otherwise, new keys found in *parameters* are ignored.

Parameters**parameters**

[dict] Parameters to update.

Notes

parameters is presumed to have been produced by the *Step.get_pars* method. As such, the “steps” key is treated special in that it is a dict whose keys are the steps assigned directly as parameters to the current step. This is standard practice for *Pipeline*-based steps.

Class Inheritance Diagram

INDICES AND TABLES

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

PYTHON MODULE INDEX

S

stpipe, 1

Symbols

`__call__()` (*stpipe.Step* method), 6

B

`build_config()` (*stpipe.Step* class method), 6

C

`call()` (*stpipe.Step* class method), 6

`class_alias` (*stpipe.Step* attribute), 5

`closeout()` (*stpipe.Step* method), 7

`correction_pars` (*stpipe.Step* attribute), 5

D

`default_output_file()` (*stpipe.Step* method), 7

`default_suffix()` (*stpipe.Step* method), 7

E

`export_config()` (*stpipe.Step* method), 7

F

`finalize_result()` (*stpipe.Step* method), 7

`from_cmdline()` (*stpipe.Step* static method), 7

`from_config_file()` (*stpipe.Step* class method), 8

`from_config_section()` (*stpipe.Step* class method), 8

G

`get_config_from_reference()` (*stpipe.Pipeline* class method), 2

`get_config_from_reference()` (*stpipe.Step* class method), 8

`get_config_reftype()` (*stpipe.Step* class method), 9

`get_pars()` (*stpipe.Pipeline* method), 2

`get_pars()` (*stpipe.Step* method), 9

`get_ref_override()` (*stpipe.Pipeline* method), 3

`get_ref_override()` (*stpipe.Step* method), 9

`get_reference_file()` (*stpipe.Step* method), 9

I

`input_dir` (*stpipe.Step* attribute), 5

L

`load_spec_file()` (*stpipe.Pipeline* class method), 3

`load_spec_file()` (*stpipe.Step* class method), 10

`log_records` (*stpipe.Step* attribute), 5

M

`make_input_path()` (*stpipe.Step* method), 10

`make_output_path` (*stpipe.Step* attribute), 5

`merge_config()` (*stpipe.Pipeline* class method), 3

`merge_config()` (*stpipe.Step* class method), 10

`merge_pipeline_config()` (*stpipe.Pipeline* class method), 3

module

`stpipe`, 1

N

`name_format` (*stpipe.Step* attribute), 5

O

`open_model()` (*stpipe.Step* method), 10

P

`Pipeline` (class in *stpipe*), 1

`prefetch()` (*stpipe.Step* method), 10

`prefetch_references` (*stpipe.Step* attribute), 5

`print_configspec()` (*stpipe.Step* class method), 10

`process()` (*stpipe.Step* method), 10

R

`reference_file_types` (*stpipe.Pipeline* attribute), 2

`reference_file_types` (*stpipe.Step* attribute), 5

`reference_uri_to_cache_path()` (*stpipe.Step* class method), 10

`remove_suffix()` (*stpipe.Step* static method), 10

`resolve_file_name()` (*stpipe.Step* method), 11

`run()` (*stpipe.Step* method), 11

S

`save_model()` (*stpipe.Step* method), 11

`search_attr()` (*stpipe.Step* method), 11

`set_input_filename()` (*stpipe.Pipeline* method), 3

`set_primary_input()` (*stpipe.Step* method), 12
`spec` (*stpipe.Pipeline* attribute), 2
`spec` (*stpipe.Step* attribute), 5
`Step` (class in *stpipe*), 3
`step_defs` (*stpipe.Pipeline* attribute), 2
`stpipe`
 module, 1

U

`update_pars()` (*stpipe.Step* method), 12
`use_correction_pars` (*stpipe.Step* attribute), 6