
bjec Documentation

Release 0.3.dev1

Seoester

May 14, 2023

Contents:

1	Getting Started	1
1.1	Example	1
1.2	Command Line Interface	1
1.3	Library Use	1
2	Runnables	3
2.1	Job	3
2.2	Build	3
3	Concepts	5
3.1	Constructor	5
3.2	Configuration File	6
3.3	Fluid Builder	6
4	API Reference	7
4.1	bjec package	7
5	Indices and tables	33
	Python Module Index	35
	Index	37

CHAPTER 1

Getting Started

1.1 Example

1.2 Command Line Interface

1.3 Library Use

CHAPTER 2

Runnables

2.1 Job

2.2 Build

3.1 Constructor

3.1.1 Constructor Function

A constructor function is similar to `__init__()` in Python. It is function which configures an object instance without being attached to the object's type.

It is suitable to alleviate the need of sub-classing: A constructor function is defined for each kind of `:obj:Job` instead of a sub-class with only a `__init__()` method. Furthermore, additional capabilities can be made available within the constructor function. This results in a well-defined interface for the user. For example, the list of dependencies is already populated. Constructor function can be defined for each kind of user-constructible object.

Similar to `self`, an object to be manipulated is passed as the first parameter. This is the *constructor object*. Constructor objects are not the final object to be configured, but instead a mutable container which holds all configurable values. For the user these details are not relevant and

The parameter is commonly named after the type. For example, `job` or `j` is used for `:obj:Job` constructor functions.

The lifecycle of object construction through a constructor function is as follows:

1. The constructor object is initialised. It may receive additional data such as the list of dependencies or be linked to an instance of the class which it configures. The constructor object types are implemented through the use of mix-ins with clear responsibilities.
2. The constructor function is called with the constructor object as its first parameter.
3. The final object, e.g. a `:obj:Job` instance, is finalised from the configuration stored in the constructor object.

These steps are performed by the constructor of the final object, e.g. the `:obj:Job` instance, during the `:obj:Runnable.run` call.

3.2 Configuration File

3.3 Fluid Builder

4.1 bjec package

4.1.1 Submodules

4.1.2 bjec.build module

```
class bjec.build.Build(constructor_func, depends=None)
    Bases:      bjec.master.Dependency, bjec.master.Constructible, bjec.master.
               Artefactor, bjec.master.WrapperRun, bjec.master.Runnable

    class Constructor(obj: Any)
        Bases:  bjec.master.ResolveConstructor, bjec.master.Constructor, bjec.
               master.Constructor

        builder(builder)

        dependencies

        source(source)

class bjec.build.Builder
    Bases: object

    build()
        Must be implemented by inheriting classes.

    last_built()
        Must be implemented by inheriting classes.

class bjec.build.ChangeInfo(status, last_changed)
    Bases: object

    Comprises information about the state of changes of a Source.

    A ChangeInfo-like object is returned by Source.scan().
```

status

Conveys any knowledge the Source has about whether changes have taken place. A Source may set `status` to `CHANGED`, when it changed its files directly, e.g. pulled from a remote source, etc. `UNCHANGED` may be set, when a version management system did not perform an update, `UNKNOWN` is the general case.

Type *ChangeInfo.Status*

last_changed

Date and time of the last change which took place in the Source. Generally only changes to a file's content are regarded as change.

Type `datetime.datetime`

class Status

Bases: `enum.Enum`

An enumeration.

CHANGED = 2

UNCHANGED = 1

UNKNOWN = 0

class `bjec.build.GitRepo(url, branch='master')`

Bases: *bjec.build.Source*

docstring for `GitRepo`

Parameters

- **url** (*str*) – Remote URL of the repository
- **branch** (*str*) – Branch of the remote repository to use, default: “master”

Configuration Options:

- **repos_path**: Path to local directory which repositories are downloaded to, defaults to *default_repos_path*
- **identity_file**: Path to an (SSH) identity file for authentication
- **identity_content**: Content of an (SSH) identity file for authentication

default_repos_path = `'~/bjec/repos'`

See configuration option `repos_path`.

local_path()

Return the (base) path to the source on the local file system.

Must be implemented by inheriting classes.

Returns The absolute path to the Source's local base directory.

Return type `str`

scan()

Perform a scan over the source set and return change info.

Must be implemented by inheriting classes.

Returns An object adhering to the `ChangeInfo` documentation.

Return type *ChangeInfo*

class `bjec.build.Local` (*path*)

Bases: `bjec.build.Source`

docstring for Local

local_path ()

Return the (base) path to the source on the local file system.

Must be implemented by inheriting classes.

Returns The absolute path to the Source's local base directory.

Return type `str`

scan ()

Perform a scan over the source set and return change info.

Must be implemented by inheriting classes.

Returns An object adhering to the ChangeInfo documentation.

Return type `ChangeInfo`

class `bjec.build.Make` (*path*, *target=None*, *creates=None*, *clean_first=False*, *clean_target=None*)

Bases: `bjec.build.Builder`

docstring for Make

Parameters

- **path** (*str*) – Path to the directory containing the Makefile
- **target** (*str or list of str, optional*) – make target(s) to execute
- **creates** (*str or list of str, optional*) – File path(s) created by make, may be absolute (starting with “/”) or relative to *path*
- **clean_first** (*bool, optional*) – When True, call *clean()* before starting to build (*clean_target* must be given)
- **clean_target** (*str or list of str, optional*) – make target(s) to execute for cleaning

Configuration Options:

- **environment**: Map of environment variables passed to the make call

build ()

Must be implemented by inheriting classes.

clean ()

last_built ()

Returns

The earliest mtime of any file in *creates*.

If *creates* is None, empty or None of the files exist, `datetime.datetime.min` (aware, i.e. with added tzinfo) is returned.

Return type `datetime.datetime`

result ()

class `bjec.build.Source`

Bases: `object`

local_path()

Return the (base) path to the source on the local file system.

Must be implemented by inheriting classes.

Returns The absolute path to the Source’s local base directory.

Return type `str`

scan()

Perform a scan over the source set and return change info.

Must be implemented by inheriting classes.

Returns An object adhering to the `ChangeInfo` documentation.

Return type `ChangeInfo`

`bjec.build.build(depends=None, master=None)`

4.1.3 bjec.cli module

class `bjec.cli.RunArgs`

Bases: `object`

`bjec.cli.main()` → `None`

`bjec.cli.run(args: bjec.cli.RunArgs)` → `None`

4.1.4 bjec.collector module

class `bjec.collector.Collector`

Bases: `typing.Generic`, `abc.ABC`

Collects and processes per-parameter set results.

Collector provides the context manager interface. Each collector is a non-reentrant context manager. Any long-held resources will only be acquired upon entering the context manager, i.e. by opening an aggregation file. These resources will be released when exiting the context manager, i.e. closing all open files.

collect (*results: Iterable[Tuple[Mapping[str, Any], _T_contra]]*) → `None`

Collects and processes all elements within *results*.

This method **must** be called while the context manager is in the open state.

Collect may be called never, once or multiple times.

Parameters *results* – Iterable over tuples of the parameter set with the associated result.

class `bjec.collector.Concatenate` (*path: Union[str, bytes, os.PathLike, None] = None, before_all: Union[bjec.io.Writable, str, bytes, None] = None, after_all: Union[bjec.io.Writable, str, bytes, None] = None, before: Union[bjec.io.Writable, str, bytes, bjec.params.ParamsEvaluable[typing.Union[bjec.io.Writable, str, bytes]]][Union[bjec.io.Writable, str, bytes], None] = None, after: Union[bjec.io.Writable, str, bytes, bjec.params.ParamsEvaluable[typing.Union[bjec.io.Writable, str, bytes]]][Union[bjec.io.Writable, str, bytes], None] = None*)

Bases: `bjec.collector.Collector`

Concatenates file-like openables into a new file.

Parameters **path** – The file path to be opened as the aggregate file. If `None` a temporary file is created (which is not deleted).

collect (*results: Iterable[Tuple[Mapping[str, Any], bjec.io.ReadOpenable]]*) → None

Collects and processes all elements within *results*.

This method **must** be called while the context manager is in the open state.

Collect may be called never, once or multiple times.

Parameters **results** – Iterable over tuples of the parameter set with the associated result.

path

```
class bjec.collector.Convert (f: Callable[[T_contra], S], collector:
                             bjec.collector.Collector[~S][S])
```

Bases: `bjec.collector.Collector`, `typing.Generic`

collect (*results: Iterable[Tuple[Mapping[str, Any], T_contra]]*) → None

Collects and processes all elements within *results*.

This method **must** be called while the context manager is in the open state.

Collect may be called never, once or multiple times.

Parameters **results** – Iterable over tuples of the parameter set with the associated result.

collector

```
class bjec.collector.Demux (keys: Iterable[str], factory: Callable[[Mapping[str, Any],
                                                                    bjec.collector.Collector[-T_contra][T_contra]])
```

Bases: `bjec.collector.Collector`, `typing.Generic`

Demux de-multiplexes results, by distributing to different Collectors.

Parameters

- **keys** – Keys in the parameter set which to consider during demuxing. For each distinct combination of values of these keys, a collector is maintained.
- **factory** – Function to call to create a new collector. A reduced parameter set is passed as the only argument, containing only those parameters specified in *keys*.

collect (*results: Iterable[Tuple[Mapping[str, Any], T_contra]]*) → None

Collects and processes all elements within *results*.

This method **must** be called while the context manager is in the open state.

Collect may be called never, once or multiple times.

Parameters **results** – Iterable over tuples of the parameter set with the associated result.

keys

```
class bjec.collector.Multi (*collectors)
```

Bases: `bjec.collector.Collector`, `typing.Generic`

collect (*results: Iterable[Tuple[Mapping[str, Any], T_contra]]*) → None

Collects and processes all elements within *results*.

This method **must** be called while the context manager is in the open state.

Collect may be called never, once or multiple times.

Parameters **results** – Iterable over tuples of the parameter set with the associated result.

collectors

```
class bjec.collector.Noop
    Bases: bjec.collector.Collector, typing.Generic

    collect (results: Iterable[Tuple[Mapping[str, Any], _T_contra]]) → None
        Collects and processes all elements within results.

        This method must be called while the context manager is in the open state.

        Collect may be called never, once or multiple times.

        Parameters results – Iterable over tuples of the parameter set with the associated result.
```

4.1.5 bjec.config module

```
class bjec.config.Config (namespace: str = 'bjec')
    Bases: object

    namespace

    read_yaml (path: Union[str, bytes, os.PathLike]) → None

    user

class bjec.config.ModuleConfig (config: bjec.config.Config, key_parts: Iterable[str])
    Bases: object

    get (key: str, default: Optional[Any] = None) → Optional[Any]

    key_parts
```

4.1.6 bjec.csv module

```
class bjec.csv.Collector (path: Union[str, bytes, os.PathLike, None] = None, before_all: Optional[Iterable[Iterable[Any]]] = None, after_all: Optional[Iterable[Iterable[Any]]] = None, before: Union[bjec.params.ParamsEvaluable[typing.Iterable[typing.Iterable[typing.Any]]][Iterable[Iterable[Iterable[Union[Iterable[Union[Any, bjec.params.ParamsEvaluable[typing.Any][Any]]], Iterable[Any], bjec.params.ParamsEvaluable[typing.Iterable[typing.Any]][Iterable[Any]]], None] = None, after: Union[bjec.params.ParamsEvaluable[typing.Iterable[typing.Iterable[typing.Iterable[Union[Iterable[Union[Any, bjec.params.ParamsEvaluable[typing.Any][Any]]], Iterable[Any], bjec.params.ParamsEvaluable[typing.Iterable[typing.Any]][Iterable[Any]]], None] = None, before_row: Union[Iterable[Union[Any, bjec.params.ParamsEvaluable[typing.Any][Any]]], Iterable[Any], bjec.params.ParamsEvaluable[typing.Iterable[typing.Any]][Iterable[Any]], None] = None, after_row: Union[Iterable[Union[Any, bjec.params.ParamsEvaluable[typing.Any][Any]]], Iterable[Any], bjec.params.ParamsEvaluable[typing.Iterable[typing.Any]][Iterable[Any]], None] = None, manage_headers: bool = False, before_header_row: Optional[Iterable[Any]] = None, after_header_row: Optional[Iterable[Any]] = None, input_encoding: Optional[str] = None, input_errors: Optional[str] = None, input_csv_args: Optional[Mapping[str, Any]] = None, output_encoding: Optional[str] = None, output_errors: Optional[str] = None, output_csv_args: Optional[Mapping[str, Any]] = None)
    Bases: bjec.collector.Collector

    Concatenates CSV from file-like read openables into an aggregate file.
```


Parameters

- **path** – The file path to be opened as the aggregate file. If `None` a file in a system specific location for temporary files is created. This file is never deleted by the Collector but may be deleted by OS mechanisms. It should not be treated as permanent.
- **before_all** – Rows (of columns) added before any rows from input files. That means these are written at the very beginning of the aggregate file. If `manage_headers` is `True`, these rows are written **after** the header row into the aggregate file.
- **after_all** – Rows (of columns) added after all rows from input files. That means these are written at the very end of the aggregate file.
- **before** – Rows (of columns) added before any input rows for each input file. That means these are written at the beginning of input file specific rows in the aggregate file. Parameters of the input file may be used in `before`.
- **after** – Rows (of columns) added after all input rows for each input file. That means these are written at the end of input file specific rows in the aggregate file. Parameters of the input file may be used in `after`.
- **before_row** – Columns inserted before each input row. That means these are written at the beginning of each input row in the aggregate file. Parameters of the input file may be used in `before_row`.
- **after_row** – Columns inserted after each input row. That means these are written at the end of each input row in the aggregate file. Parameters of the input file may be used in `after_row`.
- **manage_headers** – If `True` the first row of each input file is treated as a header row and only actual data rows in input files are concatenated. The header row is written once at the very beginning of the aggregate file. For this to work, the headers of all input file must be identical. An exception is raised if inconsistent
- **before_header_row** – Columns added before any input headers. That means these are written at the front of the header row of the aggregate file. Only interpreted if `manage_headers` is `True`.
- **after_header_row** – Columns added after any input headers. That means these are written at the end of the header row of the aggregate file. Only interpreted if `manage_headers` is `True`.
- **input_encoding** – Encoding to use when reading input files. Passed as-is to the `TextIOWrapper` constructor.
- **input_error** – Error setting to use when reading input files. Passed as-is to the `TextIOWrapper` constructor.
- **input_csv_args** – Args passed to `csv.reader()` when constructing readers for input files. This may include the `dialect` key.
- **output_encoding** – Encoding to use when writing output files. Passed as-is to the `TextIOWrapper` constructor.
- **output_error** – Error setting to use when writing output files. Passed as-is to the `TextIOWrapper` constructor.
- **output_csv_args** – Args passed to `csv.writer()` when constructing the writer for output file. This may include the `dialect` key.

collect (*results: Iterable[Tuple[Mapping[str, Any], bjec.io.ReadOpenable]]*) → `None`
 Collects and processes all elements within `results`.

This method **must** be called while the context manager is in the open state.

Collect may be called never, once or multiple times.

Parameters **results** – Iterable over tuples of the parameter set with the associated result.

path

4.1.7 bjec.generator module

```
class bjec.generator.Chain(*generators)
    Bases: bjec.generator.Generator
```

```
class bjec.generator.FromIterable(it: Iterable[Mapping[str, Any]])
    Bases: bjec.generator.Generator
```

```
class bjec.generator.Generator
    Bases: abc.ABC
```

Produces parameter sets.

A top-level generator produces fully specified parameter sets. Each such parameter set results in an independent invocation or execution when processed by a `Processor`.

So-called higher-level generators take other generators on input and combine the produced parameter sets in specific ways.

The `Generator ABC` is basically a standard python iterable, i.e. the `__iter__` method has to be defined and return an iterator.

```
class bjec.generator.Literal(**params)
    Bases: bjec.generator.Generator
```

```
class bjec.generator.Matrix(**params)
    Bases: bjec.generator.Generator
```

```
class bjec.generator.Product(*generators)
    Bases: bjec.generator.Generator
```

```
class bjec.generator.Repeat(generator: bjec.generator.Generator, n: int)
    Bases: bjec.generator.Generator
```

4.1.8 bjec.htcondor module

4.1.9 bjec.io module

```
class bjec.io.ReadOpenable(*args, **kwargs)
    Bases: typing_extensions.Protocol
```

open_bytes () → *io.BufferedIOBase*

open_text (*encoding: Optional[str] = None, errors: Optional[str] = None, newline: Optional[str] = None*) → *io.TextIOBase*

```
class bjec.io.ReadOpenableFromPath(path: Union[str, bytes, os.PathLike])
    Bases: bjec.io.ReadOpenable
```

open_bytes () → *io.BufferedIOBase*

open_text (*encoding: Optional[str] = None, errors: Optional[str] = None, newline: Optional[str] = None*) → *io.TextIOBase*

```

    path

class bjec.io.ReadOpenableWrapBinaryIO (b: BinaryIO)
    Bases: bjec.io.ReadOpenable

    open_bytes () → io.BufferedIOBase

    open_text (encoding: Optional[str] = None, errors: Optional[str] = None, newline: Optional[str] =
        None) → io.TextIOBase

class bjec.io.WriteOpenable (*args, **kwargs)
    Bases: typing_extensions.Protocol

    open_bytes () → io.BufferedIOBase

    open_text (encoding: Optional[str] = None, errors: Optional[str] = None, newline: Optional[str] =
        None) → io.TextIOBase

class bjec.io.WriteOpenableFromPath (path: Union[str, bytes, os.PathLike])
    Bases: bjec.io.WriteOpenable

    open_bytes () → io.BufferedIOBase

    open_text (encoding: Optional[str] = None, errors: Optional[str] = None, newline: Optional[str] =
        None) → io.TextIOBase

    path

class bjec.io.WriteOpenableWrapBinaryIO (b: BinaryIO)
    Bases: bjec.io.WriteOpenable

    open_bytes () → io.BufferedIOBase

    open_text (encoding: Optional[str] = None, errors: Optional[str] = None, newline: Optional[str] =
        None) → io.TextIOBase

class bjec.io.Writable (*args, **kwargs)
    Bases: typing_extensions.Protocol

    write_to (w: bjec.io.WriteOpenable) → None

class bjec.io.WritableFromBytes (content: bytes)
    Bases: bjec.io.Writable

    content

    write_to (w: bjec.io.WriteOpenable) → None

class bjec.io.WritableFromPath (path: Union[str, bytes, os.PathLike])
    Bases: bjec.io.Writable

    class Parameterised (path: Union[str, bytes, os.PathLike, bjec.params.ParamsEvaluable[typing.Union[str,
        bytes, os.PathLike]][Union[str, bytes, os.PathLike]]])
        Bases: object

        evaluate_with_params (params: Mapping[str, Any]) → bjec.io.WritableFromPath

    path

    write_to (w: bjec.io.WriteOpenable) → None

class bjec.io.WritableFromStr (content: str, encoding: Optional[str] = None, errors: Op-
    tional[str] = None, newline: Optional[str] = None)
    Bases: bjec.io.Writable

    content

    encoding

```

```
errors
newline
write_to (w: bjec.io.WriteOpenable) → None

class bjec.io.WriteableWrapFunc (func: Callable[[bjec.io.WriteOpenable], None])
  Bases: bjec.io.Writeable
  write_to (w: bjec.io.WriteOpenable) → None

bjec.io.ensure_writeable (source: Union[bjec.io.Writeable, str, bytes]) → bjec.io.Writeable

bjec.io.resolve_abs_path (path: Union[str, bytes, os.PathLike,
    bjec.params.ParamsEvaluatable[typing.Union[str, bytes,
    os.PathLike]][Union[str, bytes, os.PathLike]], params: Mapping[str,
    Any]) → Union[str, bytes]

bjec.io.resolve_path (path: Union[str, bytes, os.PathLike, bjec.params.ParamsEvaluatable[typing.Union[str,
    bytes, os.PathLike]][Union[str, bytes, os.PathLike]], params: Mapping[str,
    Any]) → Union[str, bytes]

bjec.io.resolve_writable (source: Union[bjec.io.Writeable, str, bytes,
    bjec.params.ParamsEvaluatable[typing.Union[bjec.io.Writeable, str,
    bytes]][Union[bjec.io.Writeable, str, bytes]], params: Mapping[str,
    Any]) → bjec.io.Writeable
```

4.1.10 bjec.job module

```
class bjec.job.Job (constructor_func: Callable[[Job.Constructor], None], depends: List[Union[str,
    Callable[[...], None]]] = [])
  Bases: bjec.master.Dependency, bjec.master.Constructible, bjec.master.Artefactor, bjec.master WrapperRun, bjec.master.Runnable

  class Constructor (obj: Any)
    Bases: bjec.master.ResolveConstructor, bjec.master.Constructor, bjec.master.Constructor
    after (*after_funcs) → None
    collector
    generator
    processor
    runnable

  collector
  generator
  processor
  run () → None
    Must be implemented by inheriting classes.
  runnable

bjec.job.job (depends: List[Union[str, Callable[[...], None]]] = [], master: Optional[bjec.master.Master] = None) → Callable[[Callable[[bjec.job.Job.Constructor],
    None], Callable[[...], None]]]
```

4.1.11 bjec.json module

```
class bjec.json.Writeable (value: Union[Any, bjec.params.ParamsEvaluable[typing.Any][Any], It-
    erable[Union[Any, bjec.params.ParamsEvaluable[typing.Any][Any]]],
    Iterable[Any], bjec.params.ParamsEvaluable[typing.Iterable[typing.Any]][Iterable[Any]],
    Mapping[Union[str, bjec.params.ParamsEvaluable[str][str]],
    Union[Any, bjec.params.ParamsEvaluable[typing.Any][Any]]], Map-
    ping[str, Any], bjec.params.ParamsEvaluable[typing.Mapping[str,
    typing.Any]][Mapping[str, Any]]])

Bases: object

evaluate_with_params (params: Mapping[str, Any]) → bjec.io.WriteableWrapFunc
```

4.1.12 bjec.master module

```
class bjec.master.Artefactor
    Bases: object
    docstring for Artefactor

class Constructor
    Bases: object
    add_artefacts (**kwargs) → None
    add_artefacts (**kwargs) → None
    artefacts
    w_run () → None

class bjec.master.Constructible
    Bases: object
    docstring for Constructible

class Constructor (obj: Any)
    Bases: object
    construct () → None
    constructed
    constructor_func
    w_run () → None

class bjec.master.Dependency
    Bases: bjec.master.Registerable, object
    docstring for Dependency

    Dependency has two different Constructor variants: SetUpConstructor allows adding dependencies to the
    object, while ResolveConstructor makes resolved dependencies available with its dependencies attribute.

class ResolveConstructor
    Bases: object
    dependencies

class SetUpConstructor
    Bases: object
```

depends (*args) → None

depends (*args) → None

fulfill () → None

Fulfills this dependency.

May be implemented by inheriting classes, but defaults to calling *self.run()*. In this case however, *self.run()* has to ensure *_fulfill_dependencies()* is run.

Should the object only be run once, the following can be inserted at the beginning of this method's implementation (or *self.run()*):

```
if self.fulfilled():
    return
```

fulfilled () → bool

w_run () → None

class *bjec.master.Master*

Bases: object

register (*obj*: *bjec.master.Registerable*, *func*: *Callable[[...], None]*, *aliases*: *Union[str, Sequence[str], None]* = None) → None

class *bjec.master.Registerable*

Bases: object

registered_with (*master*: *bjec.master.Master*) → None

class *bjec.master.Runnable*

Bases: *abc.ABC*

run () → None

Must be implemented by inheriting classes.

class *bjec.master WrapperRun*

Bases: object

docstring for WrapperRun

run () → None

w_run () → None

4.1.13 *bjec.params* module

class *bjec.params.Call* (*func*: *Callable[[...], _T]*, *args, **kwargs)

Bases: *bjec.params._IdentityMixin*, *bjec.params._WithMixin*, *typing.Generic*

Calls a function with *ParamsEvaluable* arguments.

Call can also be used to instantiate objects, as this happens in the same way a function is called.

Example

```
Call(Concatenate, file_path=Join("out.", P("n"), ".data"), close_files=True)
```

Parameters

- **func** – Function to be called.
- ***args** – Variable arguments passed to the class constructor. May contain `ParamsEvaluable` elements.
- ****kwargs** – Keyword arguments passed to the class constructor. May contain `ParamsEvaluable` values.

evaluate_with_params (*params: Mapping[str, Any]*) → *_T*

class `bjec.params.Dict`

Bases: `bjec.params._WithMixin`, `typing.Generic`

Utility to construct complex dictionaries depending on parameters.

Example

```
{'--mu': P('mu')} + Dict.Conditional(lambda p: 'sigma' in p, {'--sigma': P('sigma'
→)}) + {P('extra_key'): P('extra_value')}
```

```
class Conditional (condition: Callable[[Mapping[str, Any]],
                                bool],
                    m: Union[Mapping[Union[_T_inner,
bjec.params.ParamsEvaluable[~_T_inner]][_T_inner]],
Union[_S_inner,
bjec.params.ParamsEvaluable[~_S_inner]][_S_inner]],
Mapping[_T_inner,
_S_inner],
bjec.params.ParamsEvaluable[typing.Mapping[~_T_inner,
~_S_inner]][Mapping[_T_inner, _S_inner]]])
```

Bases: `bjec.params._Part`

evaluate_with_params (*params: Mapping[str, Any]*) → `Mapping[_T_inner, _S_inner]`

```
class Literal (m: Union[Mapping[Union[_T_inner, bjec.params.ParamsEvaluable[~_T_inner]][_T_inner]],
Union[_S_inner, bjec.params.ParamsEvaluable[~_S_inner]][_S_inner]],
Mapping[_T_inner, _S_inner], bjec.params.ParamsEvaluable[typing.Mapping[~_T_inner,
~_S_inner]][Mapping[_T_inner, _S_inner]]])
```

Bases: `bjec.params._Part`

evaluate_with_params (*params: Mapping[str, Any]*) → `Mapping[_T_inner, _S_inner]`

```
class Pairs (it: Union[Iterable[Union[Tuple[_T_inner, _S_inner],
bjec.params.ParamsEvaluable[typing.Tuple[~_T_inner,
~_S_inner]][Tuple[_T_inner, _S_inner]]],
Iterable[Tuple[_T_inner, _S_inner]],
bjec.params.ParamsEvaluable[typing.Iterable[typing.Tuple[~_T_inner,
~_S_inner]]][Iterable[Tuple[_T_inner, _S_inner]]]])
```

Bases: `bjec.params._Part`

evaluate_with_params (*params: Mapping[str, Any]*) → `Mapping[_T_inner, _S_inner]`

evaluate_with_params (*params: Mapping[str, Any]*) → `Dict[_T, _S]`

class `bjec.params.Join` (**args, sep: Optional[_T_sb] = None*)

Bases: `bjec.params._IdentityMixin`, `bjec.params._WithMixin`, `typing.Generic`

String / Bytes Join for lists containing `ParamsEvaluable` objects.

The type of output is determined by the type of the *sep* argument.

If the output should be a `str`, `str(.)` will be called on each list element (in **args*). If the output should be of type `bytes`, the user has to ensure that each of the list elements are of `bytes` type and that `ParamsEvaluable(.)` returns a `bytes` object.

Example

```
Join("out.", P("n"), ".csv")
```

Parameters

- ***args** – Elements to join, may be instances of ParamsEvaluable classes.
- **sep** – Separator used to join elements of **args*. Must have the type of the output, i.e. if the output should be of a bytes type, sep must be as well. Defaults to ' ' (str).

evaluate_with_params (*params: Mapping[str, Any]*) → *_T_sb*

class bjec.params.Lambda (*func: Callable[[Mapping[str, Any]], _T]*)

Bases: bjec.params._IdentityMixin, bjec.params._WithMixin, typing.Generic

Calls a function with the params dict as the only argument.

Convenient way to compute values based on parameters using a lambda expression.

Example

```
Lambda(lambda p: p['alpha'] / p['beta'])
```

Parameters **func** – Function to be called on evaluation. The params dict is passed as the only argument.

evaluate_with_params (*params: Mapping[str, Any]*) → *_T*

class bjec.params.List

Bases: bjec.params._WithMixin, typing.Generic

Utility to construct complex lists depending on parameters.

Example

```
[ '--mu', P('mu') ] + List.Conditional(lambda p: 'sigma' in p, [ '--sigma', P('sigma  
→') ] ) + [ '-']
```

class Conditional (*condition: Callable[[Mapping[str, Any]], bool], it: Union[Iterable[Union[_T_inner, bjec.params.ParamsEvaluable[~_T_inner][_T_inner]]], Iterable[_T_inner], bjec.params.ParamsEvaluable[typing.Iterable[~_T_inner]][Iterable[_T_inner]]]*)

Bases: bjec.params._Part

evaluate_with_params (*params: Mapping[str, Any]*) → Iterable[_T_inner]

class Literal (*it: Union[Iterable[Union[_T_inner, bjec.params.ParamsEvaluable[~_T_inner][_T_inner]]], Iterable[_T_inner], bjec.params.ParamsEvaluable[typing.Iterable[~_T_inner]][Iterable[_T_inner]]]*)

Bases: bjec.params._Part

evaluate_with_params (*params: Mapping[str, Any]*) → Iterable[_T_inner]

evaluate_with_params (*params: Mapping[str, Any]*) → List[_T]

class `bjec.params.P` (*key: str*)
 Bases: `bjec.params._IdentityMixin`, `bjec.params._WithMixin`, `typing.Generic`

Wrapper to allow intuitive parameter inclusion.

P instances represent a ‘future’ parameter value, every instance contains the *key* of the parameter in the *params* dict. Each instance evaluates to the corresponding parameter’s value.

Other modules may accept *P* objects or lists containing *P* objects. These are then evaluated for every parameter set.

Example

```
Environment.Fluid().set(CPUS=P('n_cpus'))
```

Parameters **key** – Parameter (key of the parameter in the *params*) dict.

evaluate_with_params (*params: Mapping[str, Any]*) → *_T*

exception `bjec.params.ParamUnavailable`

Bases: `KeyError`

classmethod **wrap_params** (*params: Mapping[str, Any]*) → *bjec.params._CustomKeyErrorMapping*
 Returns wrapped *params* raising `ParamUnavailable` on key miss.

class `bjec.params.ParamsEvaluatable` (**args, **kwargs*)

Bases: `typing_extensions.Protocol`

evaluate_with_params (*params: Mapping[str, Any]*) → *_T_co*

class `bjec.params.Path`

Bases: `bjec.params._WithMixin`

Utility to construct complex paths depending on parameters with `pathlib`.

Example

```
Path.Literal(base_dir) / P('scenario') / Path.Conditional(lambda p: 'sub_scenario'
↳ 'in p, P('sub_scenario')) / Path.Format('{case}.csv')
```

class **Conditional** (*condition: Callable[[Mapping[str, Any]], bool], s: Union[os.PathLike, str; bjec.params.ParamsEvaluatable[typing.Union[os.PathLike, str]][Union[os.PathLike, str]]]*)
 Bases: `bjec.params._Part`

evaluate_with_params (*params: Mapping[str, Any]*) → `pathlib.PurePath`

class **Format** (*format_str: str, **resolvables*)

Bases: `bjec.params._Part`

Expands a format string with the *params* dict on evaluation.

Example

```
Path.Format('{case}.csv')
```

Parameters

- **format_str** – Path which is expanded with the params dict values using `str.format()`.
- ****resolvables** – *Resolvable's which are resolved and made available as "{name}"* (the argument's name) during `str.format()` evaluation.

evaluate_with_params (*params: Mapping[str, Any]*) → `pathlib.PurePath`

class Literal (*s: Union[os.PathLike, str, bjec.params.ParamsEvaluable[typing.Union[os.PathLike, str]][Union[os.PathLike, str]]]*)
 Bases: `bjec.params._Part`

evaluate_with_params (*params: Mapping[str, Any]*) → `pathlib.PurePath`

evaluate_with_params (*params: Mapping[str, Any]*) → `pathlib.PurePath`

class bjec.params.String

Bases: `bjec.params._WithMixin`

Utility to construct complex strings depending on parameters.

Example

```
P('lambda') + String.Format('.{mu}.') + String.Conditional(lambda p: 'sigma' in p,
↳ P('sigma'))
```

class Conditional (*condition: Callable[[Mapping[str, Any]], bool], s: Union[str, bjec.params.ParamsEvaluable[str][str]]*)
 Bases: `bjec.params._Part`

evaluate_with_params (*params: Mapping[str, Any]*) → `str`

class Format (*format_str: str, **resolvables*)

Bases: `bjec.params._Part`

Expands a format string with the params dict on evaluation.

Example

```
String.Format('--nprocs={n}')
```

Parameters

- **format_str** – String which is expanded with the params dict values using `str.format()`.
- ****resolvables** – *Resolvable's which are resolved and made available as "{name}"* (the argument's name) during `str.format()` evaluation.

evaluate_with_params (*params: Mapping[str, Any]*) → `str`

class Literal (*s: Union[str, bjec.params.ParamsEvaluable[str][str]]*)
 Bases: `bjec.params._Part`

evaluate_with_params (*params: Mapping[str, Any]*) → `str`

evaluate_with_params (*params: Mapping[str, Any]*) → `str`

```
bjec.params.ensure_multi_iterable(it: Union[Iterable[Union[_T,
    bjec.params.ParamsEvaluable[~_T][_T]], Iterable[_T],
    bjec.params.ParamsEvaluable[typing.Iterable[~_T]][Iterable[_T]]])
    → Union[Iterable[Union[_T,
    bjec.params.ParamsEvaluable[~_T][_T]], Iterable[_T],
    bjec.params.ParamsEvaluable[typing.Iterable[~_T]][Iterable[_T]]]
```

Returns a multi-iterable variant of it.

An iterator is a valid iterable but can only be iterated once. This function creates a semantic copy of `it` which can be iterated many times.

If `it` fulfills the `ParamsEvaluable` protocol, it is assumed that multi-iteration is supported and `it` is returned as is. If `it` fulfills the `Sequence ABC`, multi-iteration is supported and `it` is returned as is. Otherwise, `it` is read into a list which is then returned.

```
bjec.params.resolve(obj: Union[_T, bjec.params.ParamsEvaluable[~_T][_T]], params: Mapping[str,
    Any]) → _T
```

```
bjec.params.resolve_dict(m: Union[Mapping[Union[_T, bjec.params.ParamsEvaluable[~_T][_T]],
    Union[_S, bjec.params.ParamsEvaluable[~_S][_S]], Mapping[_T, _S],
    bjec.params.ParamsEvaluable[typing.Mapping[~_T, ~_S]][Mapping[_T,
    _S]]], params: Mapping[str, Any]) → Dict[_T, _S]
```

```
bjec.params.resolve_iterable(it: Union[Iterable[Union[_T, bjec.params.ParamsEvaluable[~_T][_T]],
    Iterable[_T], bjec.params.ParamsEvaluable[typing.Iterable[~_T]][Iterable[_T]]],
    params: Mapping[str, Any]) → Iterable[_T]
```

```
bjec.params.resolve_list(it: Union[Iterable[Union[_T, bjec.params.ParamsEvaluable[~_T][_T]],
    Iterable[_T], bjec.params.ParamsEvaluable[typing.Iterable[~_T]][Iterable[_T]]],
    params: Mapping[str, Any]) → List[_T]
```

```
bjec.params.resolve_mapping(m: Union[Mapping[Union[_T, bjec.params.ParamsEvaluable[~_T][_T]],
    Union[_S, bjec.params.ParamsEvaluable[~_S][_S]], Mapping[_T, _S],
    bjec.params.ParamsEvaluable[typing.Mapping[~_T, ~_S]][Mapping[_T, _S]]],
    params: Mapping[str, Any]) → Mapping[_T, _S]
```

```
bjec.params.transform(obj: Union[_T, bjec.params.ParamsEvaluable[~_T][_T]], transform_func:
    Callable[[_T, _S]) → Union[_S, bjec.params.ParamsEvaluable[~_S][_S]]
```

4.1.14 bjec.process module

```
class bjec.process.Environment(variables: Iterable[Tuple[Union[str,
    bjec.params.ParamsEvaluable[str][str]], Union[str,
    bjec.params.ParamsEvaluable[str][str]]])
```

Bases: object

class Fluid

Bases: object

build() → bjec.process.Environment

from_environment (environment: bjec.process.Environment) → bjec.process.Environment.Fluid

inherit (blacklist: Iterable[str] = {}, whitelist: Iterable[str] = {}) → bjec.process.Environment.Fluid

set (**variables) → bjec.process.Environment.Fluid

Use `__add__` if the keys must be `ParamsEvaluable` or the variables are available as an iterable.

unset (*variables) → bjec.process.Environment.Fluid

```
unset_from_iterable (variables: Iterable[Union[str, bjec.params.ParamsEvaluable[str][str]])  
                    → bjec.process.Environment.Fluid
```

```
evaluate_with_params (params: Mapping[str, Any]) → Dict[str, str]
```

```
class bjec.process.FileAccessor (name: str, open_path: Union[str, bytes], path: Union[str, bytes,  
                                         None] = None)
```

Bases: object

Represents a file accessible for reading.

name

open_bytes () → io.BufferedIOBase

open_path

open_text (encoding: Optional[str] = None, errors: Optional[str] = None, newline: Optional[str] =
None) → io.TextIOBase

path

```
class bjec.process.Process
```

Bases: object

Process template which may contain parameter reference.

Implementer refers to a component which interprets a Process instance and executes a program accordingly for each ParamSet. It then constructs a Result which is treated as the result for each ParamSet.

Implementers should receive the information about a process execution through `Process.with_params()`. All fields are resolved and simplified as far as possible through property accessors in `Process.WithParams`.

Regarding all file related methods of the Fluid builder: Further configuration options may be made available as part of an implementer's configuration. This might include details such as the directory for temporary files, the temporary file class to use, buffering details, network transfer options, ...

Lifecycle: Construction using Fluid. Passing to processor / runner. Deferred result passing to following stages (linked to process instance? this could perform checking, e.g. is stdout available?). Process execution for each ParamSet. Construction and return of a Result instance. Finish of result processing (causes cleanup) after the following stage is done with the Result instance.

```
class FailureMode (interpret_exit_code: Union[Callable[[int], bool], NoneType] = None, inter-  
                  pret_stderr: Union[Callable[[ForwardRef('FileAccessor')], bool], NoneType]  
                  = None, interpret_stdout: Union[Callable[[ForwardRef('FileAccessor')],  
                  bool], NoneType] = None)
```

Bases: object

interpret_exit_code = None

interpret_stderr = None

interpret_stdout = None

```
class Fluid
```

Bases: object

```
add_input_file (name: str, source: Union[bjec.io.Writable, str, bytes,  
                                          bjec.params.ParamsEvaluable[typing.Union[bjec.io.Writable, str,  
                                          bytes]][Union[bjec.io.Writable, str, bytes]]], path: Union[str, bytes,  
os.PathLike, bjec.params.ParamsEvaluable[typing.Union[str, bytes,  
os.PathLike]][Union[str, bytes, os.PathLike]], None] = None, must_not_exist:  
bool = True, create_parents: bool = False, mode: int = 438,  
cleanup_after_finish: bool = False) → bjec.process.Process.Fluid
```

Adds an input file to the Process.

Parameters

- **name** – Name through which the file is available for referencing. The file’s path is available as `P('___file_NAME')` during evaluation of all `ParamsEvaluable` constructs of the `Process`. If an input file with this name already exists, its configuration is overwritten. The same name must not be used for an input file and an output file, `Process.validate()` will raise if this is the case.
- **source** – Source of the file’s content. Use `WriteableFromPath` to refer to a file in the file system.
- **path** – If not `None` the input file is made available at this path when the result is yielded. Otherwise the implementer may use a temporary file.
- **must_not_exist** – If `True` the execution is considered failed if the file already exists before the process is started. This is evaluated before the process is started and before the file is created from `source`. Only considered if `path` is not `None`, as otherwise the implementer manages the file.
- **create_parents** – If `True` all parent directories of the file are created if non-existent. Directories are created with the default mode, disregarding the `mode` parameter. Only considered if `path` is not `None`.
- **mode** – Mode bits of the file, see `os.open()` for details. Only considered when `path` is not `None` and `source` is not `None`.
- **cleanup_after_finish** – If `True` the input file is deleted when the *finish* lifetime stage is reached. Only considered when `path` is not `None`.

```
add_output_file(name: str, path: Union[str, bytes, os.PathLike,
    bjec.params.ParamsEvaluable[typing.Union[str, bytes,
    os.PathLike]]], Union[str, bytes, os.PathLike], None] = None,
    must_not_exist: bool = True, create: bool = True, create_parents:
    bool = False, mode: int = 438, cleanup_after_finish: bool = False) →
    bjec.process.Process.Fluid
```

Adds an output file to the `Process`.

Parameters

- **name** – Name through which the file is available for referencing. The file’s path is available as `P('___file_NAME')` during evaluation of all `ParamsEvaluable` constructs of the `Process`. If an output file with this name already exists, its configuration is overwritten. The same name must not be used for an input file and an output file, `Process.validate()` will raise if this is the case.
- **path** – If not `None` the output file is made available at this path when the result is yielded. Otherwise the implementer may use a temporary file.
- **must_not_exist** – If `True` the execution is considered failed if the file already exists before the process is started. This is evaluated before the process is started and before the file is created via `create`. Only considered if `path` is not `None`, as otherwise the implementer manages the file. If `False` it is considered a failure if the process did not create the file.
- **create** – If `True` the file is ensured to be present before the process is started. If `False` the file is ensured to not be present, meaning any file at the path will be deleted.
- **create_parents** – If `True` all parent directories of the file are created if non-existent. Directories are created with the default mode, disregarding the `mode` parameter. The directories are created before the process is started. Only considered if `path` is not `None`.
- **mode** – Mode bits of the file, see `os.open()` for details. Only considered when `path` is not `None`. If `create` is `True`, the bits are set before the process is started, otherwise after the process has finished.
- **cleanup_after_finish** – If `True` the output file is deleted when the *finish* lifetime stage is reached. Only considered when `path` is not `None`.

```
args(*args) → bjec.process.Process.Fluid
```

Sets the argument list with which the process is started.

Paths of input and output files are available during evaluation as `P ('__file_NAME')`.

args_from_iterable (*args: Union[Iterable[Union[str, bjec.params.ParamsEvaluable[str][str]]], Iterable[str], bjec.params.ParamsEvaluable[typing.Iterable[str]][Iterable[str]]]*)
→ `bjec.process.Process.Fluid`

Sets the argument list with which the process is started.

Paths of input and output files are available during evaluation as `P ('__file_NAME')`.

build () → `bjec.process.Process`

capture_stderr (*capture: bool = True, path: Union[str, bytes, os.PathLike, bjec.params.ParamsEvaluable[typing.Union[str, bytes, os.PathLike]]], None = None, must_not_exist: bool = True, create_parents: bool = False, mode: int = 438, cleanup_after_finish: bool = False*) → `bjec.process.Process.Fluid`

Configure whether and how stderr is captured.

Parameters

- **capture** – If `True` stderr is captured and made available in `Result` instances. Subsequent calls may disable capturing by setting `False`.
- **path** – If not `None` the stderr is made available at this path. Otherwise the implementer may use a temporary file or store content in-memory.
- **must_not_exist** – If `True` the execution is considered failed if the file already exists before the process is started. This is evaluated before the process is started. Only considered if `path` is not `None`, as otherwise the implementer manages the file.
- **create_parents** – If `True` all parent directories of the file are created if non-existent. Directories are created with the default mode, disregarding the `mode` parameter. Only considered if `path` is not `None`.
- **mode** – Mode bits of the file, see `os.open()` for details. Only considered when `path` is not `None`.
- **cleanup_after_finish** – If `True` the stderr file is deleted when the *finish* lifetime stage is reached. Only considered when `path` is not `None`.

Raises `ValueError` – If the combination of arguments is not valid.

capture_stdout (*capture: bool = True, path: Union[str, bytes, os.PathLike, bjec.params.ParamsEvaluable[typing.Union[str, bytes, os.PathLike]]], None = None, must_not_exist: bool = True, create_parents: bool = False, mode: int = 438, cleanup_after_finish: bool = False*) → `bjec.process.Process.Fluid`

Configure whether and how stdout is captured.

Parameters

- **capture** – If `True` stdout is captured and made available in `Result` instances. Subsequent calls may disable capturing by setting `False`.
- **path** – If not `None` the stdout is made available at this path. Otherwise the implementer may use a temporary file or store content in-memory.
- **must_not_exist** – If `True` the execution is considered failed if the file already exists before the process is started. This is evaluated before the process is started. Only considered if `path` is not `None`, as otherwise the implementer manages the file.
- **create_parents** – If `True` all parent directories of the file are created if non-existent. Directories are created with the default mode, disregarding the `mode` parameter. Only considered if `path` is not `None`.
- **mode** – Mode bits of the file, see `os.open()` for details. Only considered when `path` is not `None`.
- **cleanup_after_finish** – If `True` the stdout file is deleted when the *finish* lifetime stage is reached. Only considered when `path` is not `None`.

Raises `ValueError` – If the combination of arguments is not valid.

cmd (*cmd*: `Union[str, bjec.params.ParamsEvaluable[str][str]]`) → `bjec.process.Process.Fluid`
Sets the command to be executed.

The command has to be set, a process cannot execute without setting this. If unset, `Process.validate()` will raise.

How the command is resolved to a path is up to the implementer.

connect_stdin (*source*: `Union[bjec.io.Writable, str, bytes, bjec.params.ParamsEvaluable[typing.Union[bjec.io.Writable, str, bytes]][Union[bjec.io.Writable, str, bytes]], None] = None`, *path*: `Union[str, bytes, os.PathLike, bjec.params.ParamsEvaluable[typing.Union[str, bytes, os.PathLike]][Union[str, bytes, os.PathLike]], None] = None`, *must_not_exist*: `bool = True`, *create_parents*: `bool = False`, *mode*: `int = 438`, *cleanup_after_finish*: `bool = False`) → `bjec.process.Process.Fluid`

Configures a file to connect to the process's stdin.

Parameters

- **source** – Source of the file's content. Use `WritableFromPath` to refer to a file in the file system. The value `None` is the same as specifying an empty file.
- **path** – If not `None` the file is made available at this path when the result is yielded. Otherwise the implementer may use a temporary file.
- **must_not_exist** – If `True` the execution is considered failed if the file already exists before the process is started. This is evaluated before the process is started and before the file is created from `source`. Only considered if `path` is not `None`, as otherwise the implementer manages the file.
- **create_parents** – If `True` all parent directories of the file are created if non-existent. Directories are created with the default mode, disregarding the `mode` parameter. Only considered if `path` is not `None`.
- **mode** – Mode bits of the file, see `os.open()` for details. Only considered when `path` is not `None` and `source` is not `None`.
- **cleanup_after_finish** – If `True` the file is deleted when the *finish* lifetime stage is reached. Only considered when `path` is not `None`.

environment (*environment*: `Union[Mapping[Union[str, bjec.params.ParamsEvaluable[str][str]], Union[str, bjec.params.ParamsEvaluable[str][str]]], Mapping[str, str], bjec.params.ParamsEvaluable[typing.Mapping[str, str]][Mapping[str, str]]]`) → `bjec.process.Process.Fluid`

Sets the environment variables of the process.

The recommended way of constructing an environment is the `Environment` type. It can be built through a fluid interface via `Environment.Fluid`.

Paths of input and output files are available during evaluation as `P('__file_NAME')`.

failure_mode (*interpret_exit_code*: `Optional[Callable[[int], bool]] = None`, *interpret_stderr*: `Optional[Callable[[FileAccessor], bool]] = None`, *interpret_stdout*: `Optional[Callable[[FileAccessor], bool]] = None`) → `bjec.process.Process.Fluid`

Configure when a process execution is considered to be failed.

The default behaviour is to consider any execution returning a non-0 exit code as failed. If any argument is passed, this behaviour is disabled.

If *any* predicate evaluates to `True`, the execution is considered a failure.

Parameters

- **interpret_exit_code** – Predicate function to interpret the exit code. Return `True` if the exit code is considered a failure.

- **interpret_stderr** – Predicate function to interpret the stderr stream. This only works if stderr capturing is configured via `capture_stderr()`, otherwise `Process.validate()` will raise. Return True if the exit code is considered a failure.
- **interpret_stdout** – Predicate function to interpret the stdout stream. This only works if stdout capturing is configured via `capture_stdout()`, otherwise `Process.validate()` will raise. Return True if the exit code is considered a failure.

remove_input_file (*name: str*) → `bjec.process.Process.Fluid`

Removes an input file from the Process by name.

remove_output_file (*name: str*) → `bjec.process.Process.Fluid`

Removes an output file from the Process by name.

working_directory (*dir: Union[str, bjec.params.ParamsEvaluable[str][str], None]*) → `bjec.process.Process.Fluid`

Sets the working directory of the process.

If unset, implementers may execute in any directory.

class InputFile (*name: str, source: bjec.io.Writeable, path: Union[str, bytes, NoneType] = None, must_not_exist: bool = True, create_parents: bool = False, mode: int = 438, cleanup_after_finish: bool = False*)

Bases: `object`

cleanup_after_finish = `False`

create_parents = `False`

mode = `438`

must_not_exist = `True`

path = `None`

class OutputFile (*name: str, path: Union[str, bytes, NoneType] = None, must_not_exist: bool = True, create: bool = True, create_parents: bool = False, mode: int = 438, cleanup_after_finish: bool = False*)

Bases: `object`

cleanup_after_finish = `False`

create = `True`

create_parents = `False`

mode = `438`

must_not_exist = `True`

path = `None`

class Stdin (*source: Union[bjec.io.Writeable, NoneType] = None, path: Union[str, bytes, NoneType] = None, must_not_exist: bool = True, create_parents: bool = False, mode: int = 438, cleanup_after_finish: bool = False*)

Bases: `object`

cleanup_after_finish = `False`

connected

create_parents = `False`

mode = `438`


```

    must_not_exist = True

    path = None

    source = None

class Stdout (capture: bool = False, path: Union[str, bytes, NoneType] = None, must_not_exist:
    bool = True, create_parents: bool = False, mode: int = 438, cleanup_after_finish:
    bool = False)
    Bases: object

    capture = False

    cleanup_after_finish = False

    create_parents = False

    mode = 438

    must_not_exist = True

    path = None

class WithParams (process: bjec.process.Process, params: Mapping[str, Any])
    Bases: object

    args

    cmd

    environment

    failure_mode

    input_files

    output_files

    stderr

    stdin

    stdout

    working_directory

    validate () → None
        Raises if this instance is not complete or inconsistent.

    with_params (params: Mapping[str, Any]) → bjec.process.Process.WithParams

class bjec.process.Result (exit_code: int, stdin: Optional[bjec.process.FileAccessor] = None,
    stdout: Optional[bjec.process.FileAccessor] = None, stderr:
    Optional[bjec.process.FileAccessor] = None, input_files: Op-
    tional[Dict[str, bjec.process.FileAccessor]] = None, output_files:
    Optional[Dict[str, bjec.process.FileAccessor]] = None)

    Bases: object

    exit_code

    input_file (name: str) → bjec.process.FileAccessor

    output_file (name: str) → bjec.process.FileAccessor

    stderr

    stdin

    stdout

```

4.1.15 bjec.processor module

class `bjec.processor.Processor`

Bases: `typing.Generic`, `abc.ABC`

docstring for Processor

process (*runnable: Any, params_it: Iterable[Mapping[str, Any]]*) → `Iterator[Tuple[Mapping[str, Any], _T_co]]`

Process all parameter sets in the iterable according to a runnable.

Must be implemented by inheriting classes.

4.1.16 bjec.subprocessor module

class `bjec.subprocessor.FileDescriptor` (*name: str, open_path: Union[str, bytes], process_path: Union[str, bytes], temporary: bool, cleanup: bool, path: Union[str, bytes, NoneType] = None*)

Bases: `object`

accessor () → `bjec.process.FileAccessor`

path = `None`

class `bjec.subprocessor.Subprocessor` (*max_processes: int = 0*)

Bases: `bjec.processor.Processor`

Subprocessor runs Process executions concurrently using threads.

Parameters **max_processes** – Maximum number of processes to be run concurrently. If `<= 0`, the configuration option of the same name is used instead. `1` is used if the configuration option is not set.

Configuration Options:

- **max_processes: Maximum number of processes to be run** concurrently. The option is used when `max_processes` passed to the constructor is `<= 0`.

max_processes

process (*runnable: Any, params_it: Iterable[Mapping[str, Any]]*) → `Iterator[Tuple[Mapping[str, Any], bjec.process.Result]]`

Process all parameter sets in the iterable according to a runnable.

Must be implemented by inheriting classes.

`bjec.subprocessor.prepare_input_file` (*spec: Union[bjec.process.Process.InputFile, bjec.process.Process.Stdin], exit_handlers: bjec.utils.HandlersCollector, cleanup_handlers: bjec.utils.HandlersCollector, name: str = "", temp_dir: Optional[str] = None*) → `bjec.subprocessor.FileDescriptor`

`bjec.subprocessor.prepare_output_file` (*spec: Union[bjec.process.Process.OutputFile, bjec.process.Process.Stdout], exit_handlers: bjec.utils.HandlersCollector, cleanup_handlers: bjec.utils.HandlersCollector, name: str = "", temp_dir: Optional[str] = None*) → `bjec.subprocessor.FileDescriptor`

4.1.17 bjec.utils module

class `bjec.utils.CallbackOnException` (*f: Callable[[...], None], *args, **kwargs*)
 Bases: `object`

Context manager calling a function on exit only if an exception occurred.

class `bjec.utils.HandlersCollector` (**args, **kwargs*)
 Bases: `typing_extensions.Protocol`

callback (*callback: Callable[[...], Any], *args, **kwargs*) → `Callable[[...], Any]`

class `bjec.utils.HandlersList`
 Bases: `object`

callback (*callback: Callable[[...], Any], *args, **kwargs*) → `Callable[[...], Any]`

clear () → `None`

`bjec.utils.consume` (*it: Iterable[Any], n: Optional[int] = None*) → `None`
 Advance the iterable it n-steps ahead. If n is None, consume entirely.

Copied from: <https://docs.python.org/3.7/library/itertools.html#itertools-recipes>

`bjec.utils.listify` (*obj: Union[_T, Sequence[_T], None], none_empty: bool = False*) → `List[_T]`
 Turns obj into a list. Returns [obj] if it.

Returns obj is simply returned, if it already is a list. Otherwise - or if it a string - it is wrapped in a list. If none_empty is set to True, an empty list is returned, if obj is None.

`bjec.utils.max_datetime` = `datetime.datetime(9999, 12, 31, 23, 59, 59, 999999, tzinfo=datetime.timezone.utc)`
 Maximum representable datetime with timezone (“aware”) set to UTC.

`bjec.utils.min_datetime` = `datetime.datetime(1, 1, 1, 0, 0, 0, tzinfo=datetime.timezone.utc)`
 Minimum representable datetime with timezone (“aware”) set to UTC.

4.1.18 Module contents

CHAPTER 5

Indices and tables

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

b

- `bjec`, [31](#)
- `bjec.build`, [7](#)
- `bjec.cli`, [10](#)
- `bjec.collector`, [10](#)
- `bjec.config`, [12](#)
- `bjec.csv`, [12](#)
- `bjec.generator`, [14](#)
- `bjec.io`, [14](#)
- `bjec.job`, [16](#)
- `bjec.json`, [17](#)
- `bjec.master`, [17](#)
- `bjec.params`, [18](#)
- `bjec.process`, [23](#)
- `bjec.processor`, [30](#)
- `bjec.subprocessor`, [30](#)
- `bjec.utils`, [31](#)

A

accessor() (*bjec.subprocessor.FileDescriptor method*), 30
 add_artefacts() (*bjec.master.Artefactor method*), 17
 add_artefacts() (*bjec.master.Artefactor.Constructor method*), 17
 add_input_file() (*bjec.process.Process.Fluid method*), 24
 add_output_file() (*bjec.process.Process.Fluid method*), 25
 after() (*bjec.job.Job.Constructor method*), 16
 args (*bjec.process.Process.WithParams attribute*), 29
 args() (*bjec.process.Process.Fluid method*), 25
 args_from_iterable() (*bjec.process.Process.Fluid method*), 26
 Artefactor (*class in bjec.master*), 17
 Artefactor.Constructor (*class in bjec.master*), 17
 artefacts (*bjec.master.Artefactor attribute*), 17

B

bjec (*module*), 31
 bjec.build (*module*), 7
 bjec.cli (*module*), 10
 bjec.collector (*module*), 10
 bjec.config (*module*), 12
 bjec.csv (*module*), 12
 bjec.generator (*module*), 14
 bjec.io (*module*), 14
 bjec.job (*module*), 16
 bjec.json (*module*), 17
 bjec.master (*module*), 17
 bjec.params (*module*), 18
 bjec.process (*module*), 23
 bjec.processor (*module*), 30
 bjec.subprocessor (*module*), 30
 bjec.utils (*module*), 31
 Build (*class in bjec.build*), 7

build() (*bjec.build.Builder method*), 7
 build() (*bjec.build.Make method*), 9
 build() (*bjec.process.Environment.Fluid method*), 23
 build() (*bjec.process.Process.Fluid method*), 26
 build() (*in module bjec.build*), 10
 Build.Constructor (*class in bjec.build*), 7
 Builder (*class in bjec.build*), 7
 builder() (*bjec.build.Build.Constructor method*), 7

C

Call (*class in bjec.params*), 18
 callback() (*bjec.utils.HandlersCollector method*), 31
 callback() (*bjec.utils.HandlersList method*), 31
 CallbackOnException (*class in bjec.utils*), 31
 capture (*bjec.process.Process.Stdout attribute*), 29
 capture_stderr() (*bjec.process.Process.Fluid method*), 26
 capture_stdout() (*bjec.process.Process.Fluid method*), 26
 Chain (*class in bjec.generator*), 14
 CHANGED (*bjec.build.ChangeInfo.Status attribute*), 8
 ChangeInfo (*class in bjec.build*), 7
 ChangeInfo.Status (*class in bjec.build*), 8
 clean() (*bjec.build.Make method*), 9
 cleanup_after_finish (*bjec.process.Process.InputFile attribute*), 28
 cleanup_after_finish (*bjec.process.Process.OutputFile attribute*), 28
 cleanup_after_finish (*bjec.process.Process.Stdin attribute*), 28
 cleanup_after_finish (*bjec.process.Process.Stdout attribute*), 29
 clear() (*bjec.utils.HandlersList method*), 31
 cmd (*bjec.process.Process.WithParams attribute*), 29
 cmd() (*bjec.process.Process.Fluid method*), 27
 collect() (*bjec.collector.Collector method*), 10
 collect() (*bjec.collector.Concatenate method*), 11
 collect() (*bjec.collector.Convert method*), 11
 collect() (*bjec.collector.Demux method*), 11

[collect \(\) \(bjec.collector.Multi method\), 11](#)
[collect \(\) \(bjec.collector.Noop method\), 12](#)
[collect \(\) \(bjec.csv.Collector method\), 13](#)
[collector \(bjec.collector.Convert attribute\), 11](#)
[collector \(bjec.job.Job attribute\), 16](#)
[collector \(bjec.job.Job.Constructor attribute\), 16](#)
[Collector \(class in bjec.collector\), 10](#)
[Collector \(class in bjec.csv\), 12](#)
[collectors \(bjec.collector.Multi attribute\), 11](#)
[Concatenate \(class in bjec.collector\), 10](#)
[Config \(class in bjec.config\), 12](#)
[connect_stdin \(\) \(bjec.process.Process.Fluid method\), 27](#)
[connected \(bjec.process.Process.Stdin attribute\), 28](#)
[construct \(\) \(bjec.master.Constructible method\), 17](#)
[constructed \(bjec.master.Constructible attribute\), 17](#)
[Constructible \(class in bjec.master\), 17](#)
[Constructible.Constructor \(class in bjec.master\), 17](#)
[constructor_func \(bjec.master.Constructible attribute\), 17](#)
[consume \(\) \(in module bjec.utils\), 31](#)
[content \(bjec.io.WriteableFromBytes attribute\), 15](#)
[content \(bjec.io.WriteableFromStr attribute\), 15](#)
[Convert \(class in bjec.collector\), 11](#)
[create \(bjec.process.Process.OutputFile attribute\), 28](#)
[create_parents \(bjec.process.Process.InputFile attribute\), 28](#)
[create_parents \(bjec.process.Process.OutputFile attribute\), 28](#)
[create_parents \(bjec.process.Process.Stdin attribute\), 28](#)
[create_parents \(bjec.process.Process.Stdout attribute\), 29](#)

D

[default_repos_path \(bjec.build.GitRepo attribute\), 8](#)
[Demux \(class in bjec.collector\), 11](#)
[dependencies \(bjec.build.Build.Constructor attribute\), 7](#)
[dependencies \(bjec.master.Dependency.ResolveConstructor attribute\), 17](#)
[Dependency \(class in bjec.master\), 17](#)
[Dependency.ResolveConstructor \(class in bjec.master\), 17](#)
[Dependency.SetUpConstructor \(class in bjec.master\), 17](#)
[depends \(\) \(bjec.master.Dependency method\), 18](#)
[depends \(\) \(bjec.master.Dependency.SetUpConstructor method\), 17](#)
[Dict \(class in bjec.params\), 19](#)
[Dict.Conditional \(class in bjec.params\), 19](#)
[Dict.Literal \(class in bjec.params\), 19](#)

[Dict.Pairs \(class in bjec.params\), 19](#)

E

[encoding \(bjec.io.WriteableFromStr attribute\), 15](#)
[ensure_multi_iterable \(\) \(in module bjec.params\), 22](#)
[ensure_writeable \(\) \(in module bjec.io\), 16](#)
[environment \(bjec.process.Process.WithParams attribute\), 29](#)
[Environment \(class in bjec.process\), 23](#)
[environment \(\) \(bjec.process.Process.Fluid method\), 27](#)
[Environment.Fluid \(class in bjec.process\), 23](#)
[errors \(bjec.io.WriteableFromStr attribute\), 15](#)
[evaluate_with_params \(\) \(bjec.io.WriteableFromPath.Parameterised method\), 15](#)
[evaluate_with_params \(\) \(bjec.json.Writeable method\), 17](#)
[evaluate_with_params \(\) \(bjec.params.Call method\), 19](#)
[evaluate_with_params \(\) \(bjec.params.Dict method\), 19](#)
[evaluate_with_params \(\) \(bjec.params.Dict.Conditional method\), 19](#)
[evaluate_with_params \(\) \(bjec.params.Dict.Literal method\), 19](#)
[evaluate_with_params \(\) \(bjec.params.Dict.Pairs method\), 19](#)
[evaluate_with_params \(\) \(bjec.params.Join method\), 20](#)
[evaluate_with_params \(\) \(bjec.params.Lambda method\), 20](#)
[evaluate_with_params \(\) \(bjec.params.List method\), 20](#)
[evaluate_with_params \(\) \(bjec.params.List.Conditional method\), 20](#)
[evaluate_with_params \(\) \(bjec.params.List.Literal method\), 20](#)
[evaluate_with_params \(\) \(bjec.params.P method\), 21](#)
[evaluate_with_params \(\) \(bjec.params.ParamsEvaluable method\), 21](#)
[evaluate_with_params \(\) \(bjec.params.Path method\), 22](#)
[evaluate_with_params \(\) \(bjec.params.Path.Conditional method\), 21](#)
[evaluate_with_params \(\) \(bjec.params.Path.Format method\), 22](#)
[evaluate_with_params \(\) \(bjec.params.Path.Literal method\), 22](#)

`evaluate_with_params()` (*bjec.params.String method*), 22
`evaluate_with_params()` (*bjec.params.String.Conditional method*), 22
`evaluate_with_params()` (*bjec.params.String.Format method*), 22
`evaluate_with_params()` (*bjec.params.String.Literal method*), 22
`evaluate_with_params()` (*bjec.process.Environment method*), 24
`exit_code` (*bjec.process.Result attribute*), 29

F

`failure_mode` (*bjec.process.Process.WithParams attribute*), 29
`failure_mode()` (*bjec.process.Process.Fluid method*), 27
`FileAccessor` (*class in bjec.process*), 24
`FileDescriptor` (*class in bjec.subprocessor*), 30
`from_environment()` (*bjec.process.Environment.Fluid method*), 23
`FromIterable` (*class in bjec.generator*), 14
`fulfill()` (*bjec.master.Dependency method*), 18
`fulfilled()` (*bjec.master.Dependency method*), 18

G

`generator` (*bjec.job.Job attribute*), 16
`generator` (*bjec.job.Job.Constructor attribute*), 16
`Generator` (*class in bjec.generator*), 14
`get()` (*bjec.config.ModuleConfig method*), 12
`GitRepo` (*class in bjec.build*), 8

H

`HandlersCollector` (*class in bjec.utils*), 31
`HandlersList` (*class in bjec.utils*), 31

I

`inherit()` (*bjec.process.Environment.Fluid method*), 23
`input_file()` (*bjec.process.Result method*), 29
`input_files` (*bjec.process.Process.WithParams attribute*), 29
`interpret_exit_code` (*bjec.process.Process.FailureMode attribute*), 24
`interpret_stderr` (*bjec.process.Process.FailureMode attribute*), 24
`interpret_stdout` (*bjec.process.Process.FailureMode attribute*), 24

J

`Job` (*class in bjec.job*), 16

`job()` (*in module bjec.job*), 16
`Job.Constructor` (*class in bjec.job*), 16
`Join` (*class in bjec.params*), 19

K

`key_parts` (*bjec.config.ModuleConfig attribute*), 12
`keys` (*bjec.collector.Demux attribute*), 11

L

`Lambda` (*class in bjec.params*), 20
`last_built()` (*bjec.build.Builder method*), 7
`last_built()` (*bjec.build.Make method*), 9
`last_changed` (*bjec.build.ChangeInfo attribute*), 8
`List` (*class in bjec.params*), 20
`List.Conditional` (*class in bjec.params*), 20
`List.Literal` (*class in bjec.params*), 20
`listify()` (*in module bjec.utils*), 31
`Literal` (*class in bjec.generator*), 14
`Local` (*class in bjec.build*), 8
`local_path()` (*bjec.build.GitRepo method*), 8
`local_path()` (*bjec.build.Local method*), 9
`local_path()` (*bjec.build.Source method*), 9

M

`main()` (*in module bjec.cli*), 10
`Make` (*class in bjec.build*), 9
`Master` (*class in bjec.master*), 18
`Matrix` (*class in bjec.generator*), 14
`max_datetime` (*in module bjec.utils*), 31
`max_processes` (*bjec.subprocessor.Subprocessor attribute*), 30
`min_datetime` (*in module bjec.utils*), 31
`mode` (*bjec.process.Process.InputFile attribute*), 28
`mode` (*bjec.process.Process.OutputFile attribute*), 28
`mode` (*bjec.process.Process.Stdin attribute*), 28
`mode` (*bjec.process.Process.Stdout attribute*), 29
`ModuleConfig` (*class in bjec.config*), 12
`Multi` (*class in bjec.collector*), 11
`must_not_exist` (*bjec.process.Process.InputFile attribute*), 28
`must_not_exist` (*bjec.process.Process.OutputFile attribute*), 28
`must_not_exist` (*bjec.process.Process.Stdin attribute*), 28
`must_not_exist` (*bjec.process.Process.Stdout attribute*), 29

N

`name` (*bjec.process.FileAccessor attribute*), 24
`namespace` (*bjec.config.Config attribute*), 12
`newline` (*bjec.io.WritableFromStr attribute*), 16
`Noop` (*class in bjec.collector*), 11

O

`open_bytes()` (*bjec.io.ReadOpenable* method), 14
`open_bytes()` (*bjec.io.ReadOpenableFromPath* method), 14
`open_bytes()` (*bjec.io.ReadOpenableWrapBinaryIO* method), 15
`open_bytes()` (*bjec.io.WriteOpenable* method), 15
`open_bytes()` (*bjec.io.WriteOpenableFromPath* method), 15
`open_bytes()` (*bjec.io.WriteOpenableWrapBinaryIO* method), 15
`open_bytes()` (*bjec.process.FileAccessor* method), 24
`open_path` (*bjec.process.FileAccessor* attribute), 24
`open_text()` (*bjec.io.ReadOpenable* method), 14
`open_text()` (*bjec.io.ReadOpenableFromPath* method), 14
`open_text()` (*bjec.io.ReadOpenableWrapBinaryIO* method), 15
`open_text()` (*bjec.io.WriteOpenable* method), 15
`open_text()` (*bjec.io.WriteOpenableFromPath* method), 15
`open_text()` (*bjec.io.WriteOpenableWrapBinaryIO* method), 15
`open_text()` (*bjec.process.FileAccessor* method), 24
`output_file()` (*bjec.process.Result* method), 29
`output_files` (*bjec.process.Process.WithParams* attribute), 29

P

`P` (class in *bjec.params*), 20
`ParamsEvaluable` (class in *bjec.params*), 21
`ParamUnavailable`, 21
`path` (*bjec.collector.Concatenate* attribute), 11
`path` (*bjec.csv.Collector* attribute), 14
`path` (*bjec.io.ReadOpenableFromPath* attribute), 14
`path` (*bjec.io.WritableFromPath* attribute), 15
`path` (*bjec.io.WriteOpenableFromPath* attribute), 15
`path` (*bjec.process.FileAccessor* attribute), 24
`path` (*bjec.process.Process.InputFile* attribute), 28
`path` (*bjec.process.Process.OutputFile* attribute), 28
`path` (*bjec.process.Process.Stdin* attribute), 29
`path` (*bjec.process.Process.Stdout* attribute), 29
`path` (*bjec.subprocessor.FileDescriptor* attribute), 30
`Path` (class in *bjec.params*), 21
`Path.Conditional` (class in *bjec.params*), 21
`Path.Format` (class in *bjec.params*), 21
`Path.Literal` (class in *bjec.params*), 22
`prepare_input_file()` (in module *bjec.subprocessor*), 30
`prepare_output_file()` (in module *bjec.subprocessor*), 30
`Process` (class in *bjec.process*), 24
`process()` (*bjec.processor.Processor* method), 30

`process()` (*bjec.subprocessor.Subprocessor* method), 30
`Process.FailureMode` (class in *bjec.process*), 24
`Process.Fluid` (class in *bjec.process*), 24
`Process.InputFile` (class in *bjec.process*), 28
`Process.OutputFile` (class in *bjec.process*), 28
`Process.Stdin` (class in *bjec.process*), 28
`Process.Stdout` (class in *bjec.process*), 29
`Process.WithParams` (class in *bjec.process*), 29
`processor` (*bjec.job.Job* attribute), 16
`processor` (*bjec.job.Job.Constructor* attribute), 16
`Processor` (class in *bjec.processor*), 30
`Product` (class in *bjec.generator*), 14

R

`read_yaml()` (*bjec.config.Config* method), 12
`ReadOpenable` (class in *bjec.io*), 14
`ReadOpenableFromPath` (class in *bjec.io*), 14
`ReadOpenableWrapBinaryIO` (class in *bjec.io*), 15
`register()` (*bjec.master.Master* method), 18
`Registerable` (class in *bjec.master*), 18
`registered_with()` (*bjec.master.Registerable* method), 18
`remove_input_file()` (*bjec.process.Process.Fluid* method), 28
`remove_output_file()` (*bjec.process.Process.Fluid* method), 28
`Repeat` (class in *bjec.generator*), 14
`resolve()` (in module *bjec.params*), 23
`resolve_abs_path()` (in module *bjec.io*), 16
`resolve_dict()` (in module *bjec.params*), 23
`resolve_iterable()` (in module *bjec.params*), 23
`resolve_list()` (in module *bjec.params*), 23
`resolve_mapping()` (in module *bjec.params*), 23
`resolve_path()` (in module *bjec.io*), 16
`resolve_writable()` (in module *bjec.io*), 16
`Result` (class in *bjec.process*), 29
`result()` (*bjec.build.Make* method), 9
`run()` (*bjec.job.Job* method), 16
`run()` (*bjec.master.Runnable* method), 18
`run()` (*bjec.master WrapperRun* method), 18
`run()` (in module *bjec.cli*), 10
`RunArgs` (class in *bjec.cli*), 10
`runnable` (*bjec.job.Job* attribute), 16
`runnable` (*bjec.job.Job.Constructor* attribute), 16
`Runnable` (class in *bjec.master*), 18

S

`scan()` (*bjec.build.GitRepo* method), 8
`scan()` (*bjec.build.Local* method), 9
`scan()` (*bjec.build.Source* method), 10
`set()` (*bjec.process.Environment.Fluid* method), 23
`source` (*bjec.process.Process.Stdin* attribute), 29
`Source` (class in *bjec.build*), 9

[source\(\)](#) (*bjec.build.Build.Constructor method*), 7
[status](#) (*bjec.build.ChangeInfo attribute*), 7
[stderr](#) (*bjec.process.Process.WithParams attribute*), 29
[stderr](#) (*bjec.process.Result attribute*), 29
[stdin](#) (*bjec.process.Process.WithParams attribute*), 29
[stdin](#) (*bjec.process.Result attribute*), 29
[stdout](#) (*bjec.process.Process.WithParams attribute*), 29
[stdout](#) (*bjec.process.Result attribute*), 29
[String](#) (*class in bjec.params*), 22
[String.Conditional](#) (*class in bjec.params*), 22
[String.Format](#) (*class in bjec.params*), 22
[String.Literal](#) (*class in bjec.params*), 22
[Subprocessor](#) (*class in bjec.subprocessor*), 30
[WriteableFromStr](#) (*class in bjec.io*), 15
[WriteableWrapFunc](#) (*class in bjec.io*), 16
[WriteOpenable](#) (*class in bjec.io*), 15
[WriteOpenableFromPath](#) (*class in bjec.io*), 15
[WriteOpenableWrapBinaryIO](#) (*class in bjec.io*), 15

T

[transform\(\)](#) (*in module bjec.params*), 23

U

[UNCHANGED](#) (*bjec.build.ChangeInfo.Status attribute*), 8
[UNKNOWN](#) (*bjec.build.ChangeInfo.Status attribute*), 8
[unset\(\)](#) (*bjec.process.Environment.Fluid method*), 23
[unset_from_iterable\(\)](#)
 (*bjec.process.Environment.Fluid method*), 23
[user](#) (*bjec.config.Config attribute*), 12

V

[validate\(\)](#) (*bjec.process.Process method*), 29

W

[w_run\(\)](#) (*bjec.master.Artefactor method*), 17
[w_run\(\)](#) (*bjec.master.Constructible method*), 17
[w_run\(\)](#) (*bjec.master.Dependency method*), 18
[w_run\(\)](#) (*bjec.master.WrapperRun method*), 18
[with_params\(\)](#) (*bjec.process.Process method*), 29
[working_directory](#)
 (*bjec.process.Process.WithParams attribute*), 29
[working_directory\(\)](#) (*bjec.process.Process.Fluid method*), 28
[wrap_params\(\)](#) (*bjec.params.ParamUnavailable class method*), 21
[WrapperRun](#) (*class in bjec.master*), 18
[write_to\(\)](#) (*bjec.io.Writeable method*), 15
[write_to\(\)](#) (*bjec.io.WriteableFromBytes method*), 15
[write_to\(\)](#) (*bjec.io.WriteableFromPath method*), 15
[write_to\(\)](#) (*bjec.io.WriteableFromStr method*), 16
[write_to\(\)](#) (*bjec.io.WriteableWrapFunc method*), 16
[Writeable](#) (*class in bjec.io*), 15
[Writeable](#) (*class in bjec.json*), 17
[WriteableFromBytes](#) (*class in bjec.io*), 15
[WriteableFromPath](#) (*class in bjec.io*), 15
[WriteableFromPath.Parameterised](#) (*class in bjec.io*), 15