

---

# **FASTR Documentation**

***Release 3.3.1***

**Fastr contributors**

**Oct 13, 2022**



# CONTENTS

<b>1</b>	<b>FASTR Documentation</b>	<b>3</b>
1.1	Introduction	3
1.1.1	Philosophy	3
1.1.2	System overview	3
1.2	Quick start guide	5
1.2.1	Installation	5
1.2.2	Configuration	6
1.2.3	Creating a simple network	6
1.2.4	Running a Network	9
1.3	User Manual	10
1.3.1	Tools	10
1.3.2	Network	13
1.3.3	Data Flow	16
1.3.4	DataTypes	20
1.3.5	Execution	22
1.3.6	IOPlugins	22
1.3.7	Secrets	23
1.3.8	Debugging	23
1.3.9	Naming Convention	28
1.3.10	Provenance	29
1.4	Command Line Tools	29
1.4.1	fastr cat	30
1.4.2	fastr dump	30
1.4.3	fastr execute	31
1.4.4	fastr extract_argparse	31
1.4.5	fastr provenance	31
1.4.6	fastr pylint	32
1.4.7	fastr report	32
1.4.8	fastr run	33
1.4.9	fastr sink	33
1.4.10	fastr source	33
1.4.11	fastr test	34
1.4.12	fastr trace	35
1.4.13	fastr upgrade	36
1.4.14	fastr verify	36
1.5	Resource File Formats	37
1.5.1	Config file	37
1.5.2	Tool description	41
1.6	Plugin Reference	42
1.6.1	CollectorPlugin Reference	42

1.6.2	ExecutionPlugin Reference . . . . .	44
1.6.3	FlowPlugin Reference . . . . .	47
1.6.4	IOPlugin Reference . . . . .	47
1.6.5	Interface Reference . . . . .	53
1.6.6	ReportingPlugin Reference . . . . .	55
1.6.7	Target Reference . . . . .	56
1.7	Development and Design Documentation . . . . .	58
1.7.1	Sample flow in Fastr . . . . .	58
1.7.2	Network Execution . . . . .	61
1.7.3	Secrets . . . . .	64
1.8	Changelog . . . . .	65
1.8.1	3.3.1 - 2022-10-13 . . . . .	65
1.8.2	3.3.0 - 2021-06-11 . . . . .	66
1.8.3	3.2.3 - 2020-06-25 . . . . .	66
1.8.4	3.2.2 - 2020-06-25 . . . . .	66
1.8.5	3.2.1 - 2020-06-22 . . . . .	67
1.8.6	3.2.0 - 2020-06-19 . . . . .	67
1.8.7	3.1.4 - 2020-06-10 . . . . .	67
1.8.8	3.1.3 - 2019-11-28 . . . . .	67
1.8.9	3.1.2 - 2019-06-18 . . . . .	67
1.8.10	3.1.1 - 2019-05-02 . . . . .	68
1.8.11	3.1.0 - 2019-05-02 . . . . .	68
1.8.12	3.0.1 - 2019-03-28 . . . . .	69
1.8.13	3.0.0 - 2019-03-05 . . . . .	69
1.8.14	2.1.2 - 2018-10-24 . . . . .	70
1.8.15	2.1.1 - 2018-06-29 . . . . .	70
1.8.16	2.1.0 - 2018-04-13 . . . . .	70
1.8.17	2.0.1 - 2017-10-19 . . . . .	71
1.8.18	2.0.0 - 2017-09-28 . . . . .	71
1.8.19	1.2.2 - 2017-08-24 . . . . .	72
1.8.20	1.2.1 - 2017-04-04 . . . . .	72
1.8.21	1.2.0 - 2017-03-15 . . . . .	73
1.8.22	1.1.2 - 2016-12-22 . . . . .	74
1.8.23	1.1.1 - 2016-12-22 . . . . .	74
1.8.24	1.1.0 - 2016-12-08 . . . . .	74
<b>2</b>	<b>FASTR User reference . . . . .</b>	<b>77</b>
2.1	Fastr User Reference . . . . .	77
<b>3</b>	<b>FASTR Developer Module reference . . . . .</b>	<b>85</b>
3.1	fastr Package . . . . .	85
3.1.1	fastr Package . . . . .	85
3.1.2	exceptions Module . . . . .	86
3.1.3	globals Module . . . . .	94
3.1.4	version Module . . . . .	94
3.1.5	Subpackages . . . . .	95
<b>4</b>	<b>Indices and tables . . . . .</b>	<b>311</b>
	<b>Python Module Index . . . . .</b>	<b>313</b>
	<b>Index . . . . .</b>	<b>315</b>

FASTR is a framework that helps creating workflows of different tools. The workflows created in FASTR are automatically enhanced with flexible data input/output, execution options (local, cluster, etc) and solid provenance.

We chose to create tools by creating wrappers around executables and connecting everything with Python.

Fastr is open-source (licensed under the Apache 2.0 license) and hosted on gitlab at <https://gitlab.com/radiology/infrastructure/fastr>

For support, go to <https://groups.google.com/d/forum/fastr-users>

To get yourself a copy, see the *Installation*

The official documentation can be found at [fastr.readthedocs.io](http://fastr.readthedocs.io)

The Fastr workflow system is presented in the following article:

Hakim Achterberg, Marcel Koek, and Wiro Niessen. “Fastr: a workflow engine for advanced data flows in medical image analysis.” *Frontiers in ICT* 3 (2016): 15.

Fastr is made possible by contributions from the following people: Hakim Achterberg, Marcel Koek, Adriaan Versteeg, Thomas Phil, Mattias Hansson, Baldur van Lew, Marcel Zwiers, and Coert Metz



## FASTR DOCUMENTATION

### 1.1 Introduction

Fastr is a system for creating workflows for automated processing of large scale data. A processing workflow might also be called a processing pipeline, however we feel that a pipeline suggests a linear flow of data. Fastr is designed to handle complex flows of data, so we prefer to use the term network. We see the workflow as a network of processing tools, through which the data will flow.

The original authors work in a medical image analysis group at Erasmus MC. They often had to run analysis that used multiple programs written in different languages. Every time a experiment was set up, the programs had to be glued together by scripts (often in bash or python).

At some point the authors got fed up by doing these things again and again, and so decided to create a flexible, powerful scripting base to easily create these scripts. The idea evolved to a framework in which the building blocks could be defined in XML and the networks could be constructed in very simple scripts (similar to creating a GUI).

#### 1.1.1 Philosophy

Researchers spend a lot of time processing data. In image analysis, this often includes using multiple tools in succession and feeding the output of one tool to the next. A significant amount of time is spent either executing these tools by hand or writing scripts to automate this process. This process is time consuming and error-prone. Considering all these tasks are very similar, we wanted to write one elaborate framework that makes it easy to create pipelines, reduces the risk of errors, generates extensive logs, and guarantees reproducibility.

The Fastr framework is applicable to multiple levels of usage: from a single researcher who wants to design a processing pipeline and needs to get reproducible results for publishing; to applying a consolidated image processing pipeline to a large population imaging study. On all levels of application the pipeline provenance and managed execution of the pipeline enables you to get reliable results.

#### 1.1.2 System overview

There are a few key requirements for the design of the system:

- Any tool that your computer can run using the command line (without user interaction) should be usable by the system without modifying the tool.
- The creation of a workflow should be simple, conceptual and require no real programming.
- Networks, once created, should be usable by anyone like a simple program. All processing should be done automatically.
- All processing of the network should be logged extensively, allowing for complete reproducibility of the system (guaranteeing data provenance).

Using these requirements we define a few key elements in our system:

- A `fastr.Tool` is a definition of any program that can be used as part of a pipeline (e.g. a segmentation tool)
- A `fastr.Node` is a single operational step in the workflow. This represents the execution of a `fastr.Tool`.
- A `fastr.Link` indicates how the data flows between nodes.
- A `fastr.Network` is an object containing a collection of `fastr.Node` and `fastr.Link` that form a workflow.

With these building blocks, the creation of a pipeline will boil down to just specifying the steps in the pipeline and the flow of the data between them. For example a simple neuro-imaging pipeline could look like:

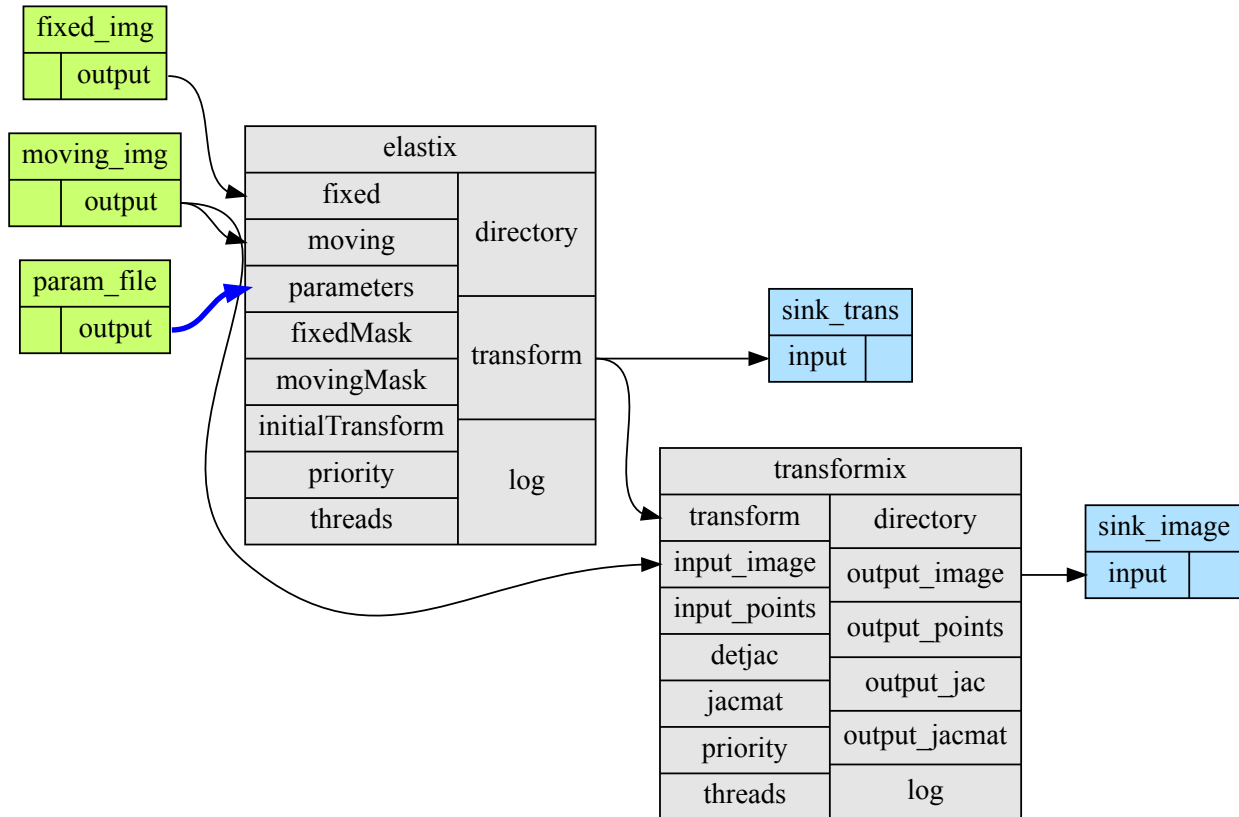


Fig. 1.1: A simple workflow that registers two images and uses the resulting transform to resample the moving image.

In Fastr this translates to:

- Create a `fastr.Network` for your pipeline
- Create a `fastr.SourceNode` for the fixed image
- Create a `fastr.SourceNode` for the moving image
- Create a `fastr.SourceNode` for the registration parameters
- Create a `fastr.Node` for the registration (in this case `elastix`)
- Create a `fastr.Node` for the resampling of the image (in this case `transformix`)
- Create a `fastr.SinkNode` to save the transformations
- Create a `fastr.SinkNode` to save the transformed images
- `fastr.Link` the output of the fixed image source node to the fixed image input of the registration node



- `fastr.Link` the output of the moving image source node to the moving image input of the registration node
- `fastr.Link` the output of the registration parameters source node to the registration parameters input of the registration node
- `fastr.Link` the output transform of the registration node to the transform input of the resampling node
- `fastr.Link` the output transform of the registration node to the input of transformation SinkNode
- `fastr.Link` the output image of the resampling node to the input of image SinkNode
- Run the `fastr.Network` for subjects X

This might seem like a lot of work for a registration, but the Fastr framework manages all other things, executes the pipeline and builds a complete paper trail of all executed operations. The execution can be on any of the supported execution environments (local, cluster, etc). The data can be imported from and exported to any of the supported data connections (file, XNAT, etc). It is also important to keep in mind that this is a simple example, but for more complex pipelines, managing the workflow with Fastr will be easier and less error-prone than writing your own scripts.

## 1.2 Quick start guide

This manual will show users how to install Fastr, configure Fastr, construct and run simple networks, and add tool definitions.

### 1.2.1 Installation

You can install Fastr either using pip, or from the source code.

#### Installing via pip

You can simply install fastr using pip:

```
pip install fastr
```

---

**Note:** You might want to consider installing `fastr` in a [virtualenv](#)

---

#### Installing from source code

To install from source code, use Mercurial via the command-line:

```
git clone https://gitlab.com/radiology/infrastructure/fastr.git # for http
git clone git@gitlab.com:radiology/infrastructure/fastr.git # for ssh
```

If you prefer a GUI you can try [TortoiseGIT](#) (Windows, Linux and Mac OS X) or [SourceTree](#) (Windows and Mac OS X). The address of the repository is (given for both http and ssh):

```
https://gitlab.com/radiology/infrastructure/fastr.git
git@gitlab.com:radiology/infrastructure/fastr.git
```

To install to your current Python environment, run:

```
cd fastr/
pip install .
```

This installs the scripts and packages in the default system folders. For windows this is the python `site-packages` directory for the fastr python library and `Scripts` directory for the executable scripts. For Ubuntu this is in the `/usr/local/lib/python3.x/dist-packages/` and `/usr/local/bin/` respectively.

**Note:** If you want to develop fastr, you might want to use `pip install -e .` to get an editable install

**Note:** You might want to consider installing fastr in a [virtualenv](#)

**Note:**

- On windows python and the `Scripts` directory are not on the system `PATH` by default. You can add these by going to `System -> Advanced Options -> Environment variables`.
- On mac you need the Xcode Command Line Tools. These can be installed using the command `xcode-select --install`.

## 1.2.2 Configuration

Fastr has defaults for all settings so it can be run out of the box to test the examples. However, when you want to create your own Networks, use your own data, or use your own Tools, it is required to edit your config file.

Fastr will search for a config file named `config.py` in the `$FASTRHOME` directory (which defaults to `~/fastr/` if it is not set). So if `$FASTRHOME` is set the `~/fastr/` will be ignored.

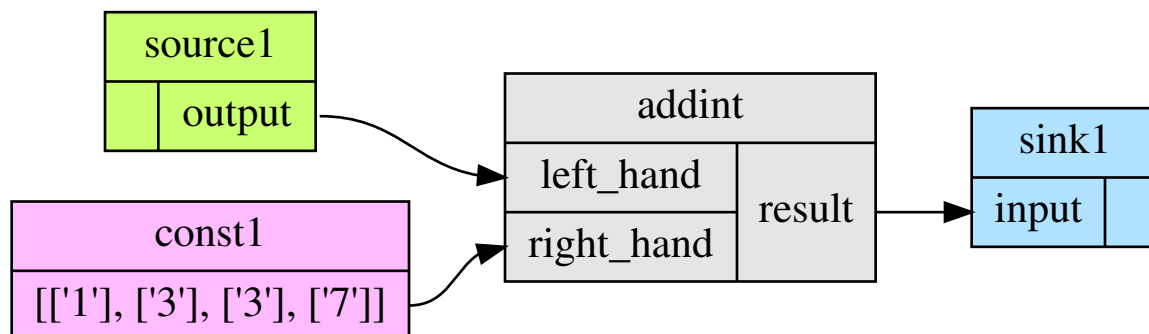
For a sample configuration file and a complete overview of the options in `config.py` see the [Config file](#) section.

## 1.2.3 Creating a simple network

If Fastr is properly installed and configured, we can start creating networks. Creating a network is very simple:

```
>>> import fastr
>>> network = fastr.create_network(id='example', version='1.0')
```

Now we have an empty network, the next step is to create some nodes and links. Imagine we want to create the following network:



## Creating nodes

We will create the nodes and add them to the network. This is done via the network `create_` methods. Let's create two source nodes, one normal node, and one sink:

```
>>> source1 = network.create_source('Int', id='source1')
>>> sink1 = network.create_sink('Int', id='sink1')
>>> addint = network.create_node('fastr/math/AddInt:1.0', tool_version='1.0', id='addint'
↪')
```

The functions `Network.create_source`, `Network.create_sink` and `Network.create_node` create the desired node and add it into the Network.

A `SourceNode` and `SinkNode` only require the datatype to be specified. A `Node` requires a `Tool` to be instantiated from. The `id` option is optional for all four, but makes it easier to identify the nodes and read the logs. The tool is defined by a namespace, the `id` and the version of the command. Many packages have multiple version which are available. The `tool_version` argument reflects the version of the Fastr wrapper which describes how the command can be called. For reproducibility also these are checked as they might be updated as well.

There is an easy way to add a constant to an input, by using a shortcut method. If you assign a `list` or `tuple` to an item in the input list, it will automatically create a `ConstantNode` and a `Link` between the `ConstantNode` and the given `Input`:

```
>>> [1, 3, 3, 7] >> addint.inputs['right_hand']
Link link_0 (network: example):
    fastr:///networks/example/1.0/nodelist/const__addint__right_hand/outputs/output ==>↪
↪fastr:///networks/example/1.0/nodelist/addint/inputs/right_hand/0
```

The created constant would have the id `const_addint__right_hand_0` as it automatically names the new constant `const_$nodeid__$inputid_$number`.

---

**Note:** The use of the `>>`, `<<`, and `=` operators for linking is discussed bellow in section [Creating links](#).

---

In an interactive python session we can simply look at the basic layout of the node using the `repr` function. Just type the name of the variable holding the node and it will print a human readable representation:

```
>>> source1
SourceNode source1 (tool: Source:1.0 v1.0)
      Inputs      |      Outputs
-----
                        | output  (Int)

>>> addint
Node addint (tool: AddInt:1.0 v1.0)
      Inputs      |      Outputs
-----
left_hand (Int)   | result  (Int)
right_hand (Int)  |
```

This tool has inputs of type `Int`, so the sources and sinks need to have a matching datatype.

The tools and datatypes available are stored in `fastr.tools` and `fastr.types`. These variables are created when `fastr` is imported for the first time. They contain all the datatype and tools specified by the `yaml`, `json` or `xml` files in the search paths. To get an overview of the tools and datatypes loaded by `fastr`:

```
>>> fastr.tools
ToolManager
...
fastr/math/Add:1.0      1.0 : ...fastr...resources...tools...fastr...math...1.0...add.
↪yaml
fastr/math/AddInt:1.0   1.0 : ...fastr...resources...tools...fastr...math...1.0...
↪addint.yaml
...

>>> fastr.types
DataTypeManager
...
Directory              : <URLType: Directory>
...
Float                  : <ValueType: Float>
...
Int                    : <ValueType: Int>
...
String                 : <ValueType: String>
...
```

The `fastr.tools` variable contains all tools that Fastr could find during initialization. Tools can be chosen in two ways:

- `tools[id]` which returns the newest version of the tool
- `tools[id, version]` which returns the specified version of the tool

## Creating links

So now we have a network with 4 nodes defined, however there is no relation between the nodes yet. For this we have to create some links.

```
>>> link1 = source1.output >> addint.inputs['left_hand']
>>> link2 = sink1.inputs['input'] << addint.outputs['result']
```

This asks the network to create links and immediately store them inside the network. A link always points from an Output to an Input (note that SubOutput or SubInputs are also valid). A SourceNode has only 1 output which is fixed, so it is easy to find. However, addImage has two inputs and one output, this requires us to specify which output we need. A normal node has a mapping with Inputs and one with Outputs. They can be indexed with the appropriate id's. The function returns the links, but you only need that if you are planning to change the properties of a link.

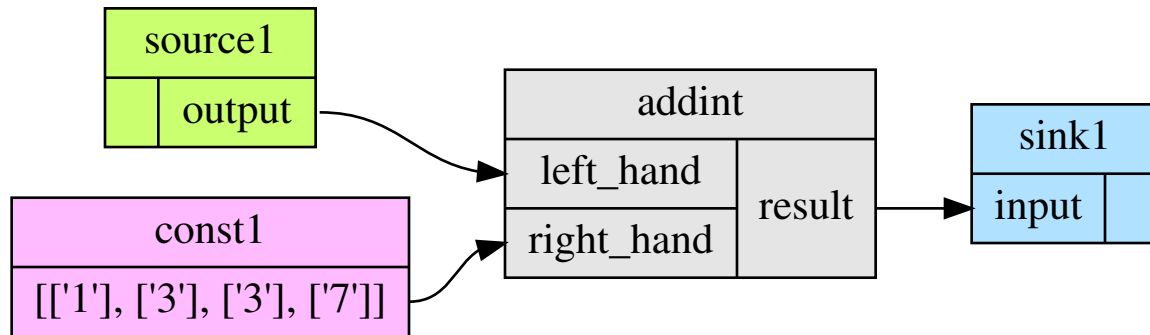
The operators with `>>` and `<<` clearly indicate the direction of the desired data flow. Also they return the created link, which is easy if you want to change the flow in a link later on. The last short hand uses the assignment, but it cannot return the created link and changing the link later on is more difficult.

## Create an image of the Network

For checking your Network it is very useful to have a graphical representation of the network. This can be achieved using the `Network.draw` method.

```
>>> network.draw()
'example.svg'
```

This will create a figure in the path returned by the function that looks like:



**Note:** for this to work you need to have graphviz installed

## 1.2.4 Running a Network

Running a network locally is almost as simple as calling the `Network.execute` method:

```
>>> source_data = {'source1': {'s1': 4, 's2': 5, 's3': 6, 's4': 7}}
>>> sink_data = {'sink1': 'vfs://tmp/fastr_result_{sample_id}.txt'}
>>> run = network.execute(source_data, sink_data)
# Lots output will appear on the stdout while running
# Show if the run was successful or if errors were encountered
>>> run.result
True
```

As you can see the execute method needs data for the sources and sinks. This has to be supplied in two `dict` that have keys matching every source/sink id in the network. Not supplying data for every source and sink will result in an error, although it is possible to pass an empty `list` to a source.

**Note:** The values of the source data have to be simple values or urls and values of the sink data have to be url templates. To see what url schemes are available and how they work see *IOPlugin Reference*. For the sink url templates see `SinkNode.set_data`

For source nodes you can supply a `list` or a `dict` with values. If you supply a `dict` the keys will be interpreted as sample ids and the values as the corresponding values. If you supply a `list`, keys will be generated in the form of `id_{N}` where N will be index of the value in the list.

**Warning:** As a `dict` does not have a fixed order, when a `dict` is supplied the samples are ordered by key to get a fixed order! For a `list` the original order is retained.

For the sink data, an url template has to be supplied that governs how the data is stored. The mini-language (the replacement fields) are described in the `SinkNode.set_data` method.

To rerun a stopped/crashed pipeline check the user manual on [Continuing a Network](#)

## 1.3 User Manual

In this chapter we will discuss the parts of Fastr in more detail. We will give a more complete overview of the system and describe the more advanced features.

### 1.3.1 Tools

The Tool in Fastr are the building blocks of each workflow. A tool represents a program/script/binary that can be called by Fastr and can be seen as a template. A *Node* can be created based on a Tool. A Node will be one processing step in a workflow, and the tool defines what the step does.

On the import of Fastr, all available Tools will be loaded in a default ToolManager that can be accessed via `fastr.tools`. To get an overview of the tools in the system, just print the `repr()` of the ToolManager:

```
>>> import fastr
>>> fastr.tools
ToolManager
...
fastr.math.Add          v0.1 : .../fastr/resources/tools/fastr/math/0.1/add.xml
fastr.math.AddInt       v0.1 : .../fastr/resources/tools/fastr/math/0.1/addint.xml
...
```

As you can see it gives the tool id, version and the file from which it was loaded for each tool in the system. To view the layout of a tool, just print the `repr()` of the Tool itself.

```
>>> fastr.tools['AddInt']
Tool AddInt v0.1 (Add two integers)
      Inputs      |      Outputs
-----
left_hand (Int)   | result  (Int)
right_hand (Int)  |
```

To add a Tool to the system a file should be added to one of the path in `fastr.config.tools_path`. The structure of a tool file is described in [Tool description](#)

### Create your own tool

There are 4 steps in creating a tool:

1. **Create folders.** We will call the tool ThrowDie. Create the folder `throw_die` in the folder `fastr-tools`. In this folder create another folder called `bin`.
2. **Place executable in correct place.** In this example we will use a snippet of executable python code:

```
#!/usr/bin/env python
import sys
import random
```

(continues on next page)

(continued from previous page)

```
import json

if (len(sys.argv) > 1):
    sides = int(sys.argv[1])
else:
    sides = 6
result = [int(random.randint(1, sides))]

print('RESULT={}'.format(json.dumps(result)))
```

Save this text in a file called `throw_die.py`

Place the executable python script in the folder `throw_die/bin`

3. **Create and edit xml file for tool.** See *tool definition reference* for all the fields that can be defined in a tool.

Put the following text in file called `throw_die.xml`.

```
<tool id="ThrowDie" description="Simulates a throw of a die. Number of sides of the
↳die is provided by user"
    name="throw_die" version="1.0">
  <authors>
    <author name="John Doe" />
  </authors>
  <command version="1.0" >
    <authors>
      <author name="John Doe" url="http://a.b/c" />
    </authors>
    <targets>
      <target arch="*" bin="throw_die.py" interpreter="python" os="*" paths='bin/' />
    </targets>
    <description>
      throw_die.py number_of_sides
      output = simulated die throw
    </description>
  </command>
  <interface>
    <inputs>
      <input cardinality="1" datatype="Int" description="Number of die sides" id=
↳"die_sides" name="die sides" nspace="False" order="0" required="True"/>
    </inputs>
    <outputs>
      <output id="output" name="output value" datatype="Int" automatic="True"
↳cardinality="1" method="json" location="^RESULT=(.*)$" />
    </outputs>
  </interface>
</tool>
```

Put `throw_die.xml` in the folder `example_tool`. All Attributes in the example above are required. For a complete overview of the xml Attributes that can be used to define a tool, check the *Tool description*. The most important Attributes in this xml are:

```
id      : The id is used in in FASTR to create an instance of your tool, this name
↳will appear in the tools when you type fastr.tools.
```

(continues on next page)

(continued from previous page)

```

targets : This defines where the executables are located and on which platform they
↳are available.
inputs  : This defines the inputs that you want to be used in FASTR, how FASTR
↳should use them and what data is allowed to be put in there.

```

More xml examples can be found in the fastr-tools folder.

- 4) **Edit configuration file.** Append the line [PATH TO LOCATION OF FASTR-TOOLS]/fastr-tools/throw\_die/ to the the config.py (located in ~/.fastr/ directory) to the tools\_path. See [Config file](#) for more information on configuration.

You should now have a working tool. To test that everything is ok do the following in python:

```

>>> import fastr
>>> fastr.tools
...

```

Now a list of available tools should be produced, including the tool ThrowDie

To test the tool create the script test\_throwdie.py:

```

import fastr

# Create network
network = fastr.create_network('ThrowDie')

# Create nodes
source1 = network.create_source('Int', id='source1')
sink1 = network.create_sink('Int', id='sink1')
throwdie = network.create_node('ThrowDie', id='throwdie')

# Create links
link1 = source1.output >> throwdie.inputs['die_sides']
link2 = throwdie.outputs['output'] >> sink1.inputs['input']

# Draw and execute
source_data = {'source1': {'s1': 4, 's2': 5, 's3': 6, 's4': 7}}
sink_data = {'sink1': 'vfs://tmp/fastr_result_{sample_id}.txt'}
network.draw()
network.execute(source_data, sink_data)

```

Call the script from commandline by

```
$ python test_throwdie.py
```

An image of the network will be created in the current directory and result files will be put in the tmp directory. The result files are called fastr\_result\_s1.txt, fastr\_result\_s2.txt, fastr\_result\_s3.txt, and fastr\_result\_s4.txt

**Note:** If you have code which is operating system depend you will have to edit the xml file. The following gives an example of how the elastix tool does this:

```

<targets>
  <target os="windows" arch="*" bin="elastix.exe">

```

(continues on next page)



(continued from previous page)

```

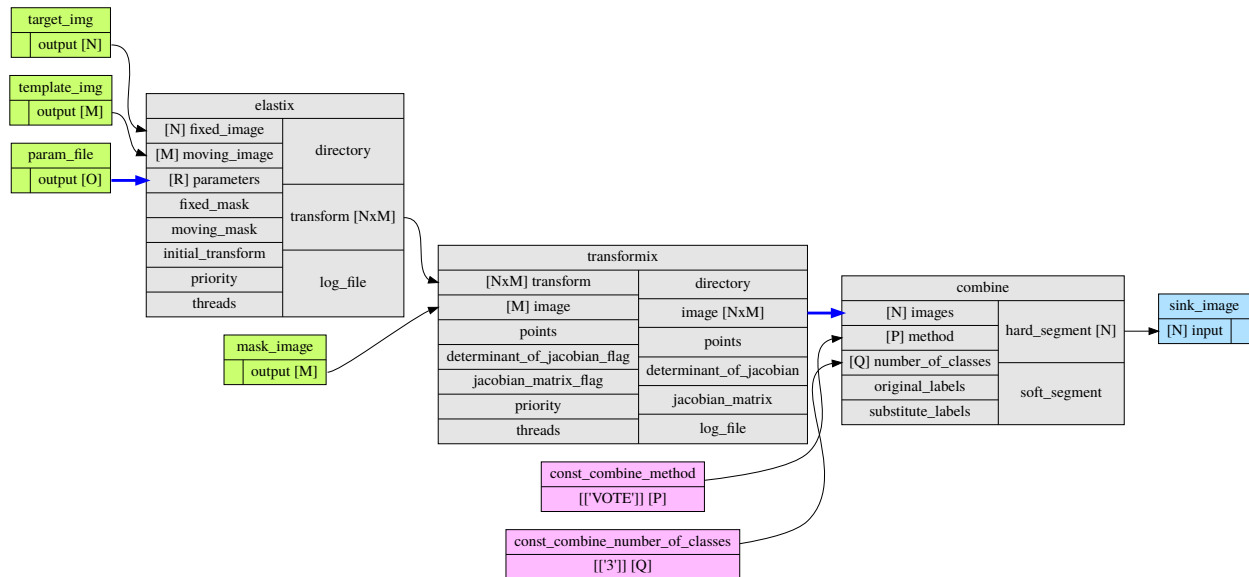
<paths>
  <path type="bin" value="vfs://apps/elastix/4.7/install/" />
  <path type="lib" value="vfs://apps/elastix/4.7/install/lib" />
</paths>
</target>
<target os="linux" arch="" modules="elastix/4.7" bin="elastix">
  <paths>
    <path type="bin" value="vfs://apps/elastix/4.7/install/" />
    <path type="lib" value="vfs://apps/elastix/4.7/install/lib" />
  </paths>
</target>
<target os="darwin" arch="" modules="elastix/4.7" bin="elastix">
  <paths>
    <path type="bin" value="vfs://apps/elastix/4.7/install/" />
    <path type="lib" value="vfs://apps/elastix/4.7/install/lib" />
  </paths>
</target>
</targets>

```

vfs is the virtual file system path, more information can be found at [VirtualFileSystem](#).

### 1.3.2 Network

A *Network* represented an entire workflow. It hold all *Nodes*, *Links* and other information required to execute the workflow. Networks can be visualized as a number of building blocks (the Nodes) and links between them:



An empty network is easy to create, all you need is to name it:

```
>>> network = fastr.create_network(id="network_name")
```

the *network* is the main interface to fastr, from it you can create all elements to create a workflow. in the following sections the different elements of a *network* will be described in more detail.

## Node

*Nodes* are the point in the *Network* where the processing happens. A *Node* takes the input data and executes jobs as specified by the underlying Tool. A *Nodes* can be created easily:

```
>>> node2 = network.create_node(tool, id='node1', step_id='step1')
```

We tell the *Network* to create a *Node* using the `create_node` method. Optionally you can add define a `step_id` for the node which is a logical grouping of *Nodes* that is mostly used for visualization.

---

**Note:** For a *Node*, the tool can be given both as the Tool class or the id of the tool. This id can be just the id or a tuple with the id and version.

---

A *Node* contains *Inputs* and *Outputs*. To see the layout of the *Node* one can simply look at the `repr()`.

```
>>> addint = network.create_node('AddInt', id='addint')
>>> addint
Node addint (tool: AddInt v1.0)
      Inputs      |      Outputs
-----
left_hand  (Int)  | result   (Int)
right_hand (Int)  |
```

The inputs and outputs are located in mappings with the same name:

```
>>> addint.inputs
<Input map, items: ['left_hand', 'right_hand']>

>>> addint.outputs
<Output map, items: ['result']>
```

The *InputMap* and *OutputMap* are classes that behave like mappings. The *InputMap* also facilitates the linking shorthand. By assigning an *Output* to an existing key, the *InputMap* will create a *Link* between the Input and Output.

## SourceNode

A *SourceNode* is a special kind of node that is the start of a workflow. The *SourceNodes* are given data at run-time that fetched via *IOPlugins*. On create, only the datatype of the data that the *SourceNode* supplied needs to be known. Creating a *SourceNode* is very similar to an ordinary node:

```
>>> source1 = network.create_source('Int', id='source1', step_id='step1', node_group=
↳ 'subject')
```

The first argument is the type of data the source supplies. The other optional arguments are for naming and grouping of the nodes. A *SourceNode* only has a single output which has a short-cut access via `source.output`.

---

**Note:** For a source or constant node, the datatype can be given both as the *BaseDataType* class or the id of the datatype.

---

## ConstantNode

A ConstantNode is another special node. It is a subclass of the SourceNode and has a similar function. However, instead of setting the data at run-time, the data of a constant is given at creation and saved in the object. Creating a ConstantNode is similar as creating a source, but with supplying data:

```
>>> constant1 = network.create_constant('Int', [42], id='constant1', step_id='step1',
↳ node_group='subject')
```

The first argument is the datatype the node supplies, similar to a SourceNode. The second argument is the data that is contained in the ConstantNode. Often, when a ConstantNode is created, it is created specifically for one input and will not be reused. In this case there is a shorthand to create and link a constant to an input:

```
>>> link = addint.inputs['value1'] << [42]
>>> link = [42] >> addint.inputs['value1']
>>> addint.inputs['value1'] = [42]
```

are three methods that will create a constant node with the value 42 and create a link between the output and input `addint.value1`.

## SinkNode

The SinkNode is the counter-part of the source node. Instead of get data into the workflow, it saves the data resulting from the workflow. For this a rule has to be given at run-time that determines where to store the data. The information about how to create such a rule is described at `SinkNode.set_data`. At creation time, only the datatype has to be specified:

```
>>> sink2 = network.create_sink('Int', id='sink2', step_id='step1', node_group='subject')
```

## Link

*Links* indicate how the data flows between *Nodes*. Links can be created explicitly using one of the following:

```
>>> link = network.create_link(node1.outputs['image'], node2.inputs['image'])
```

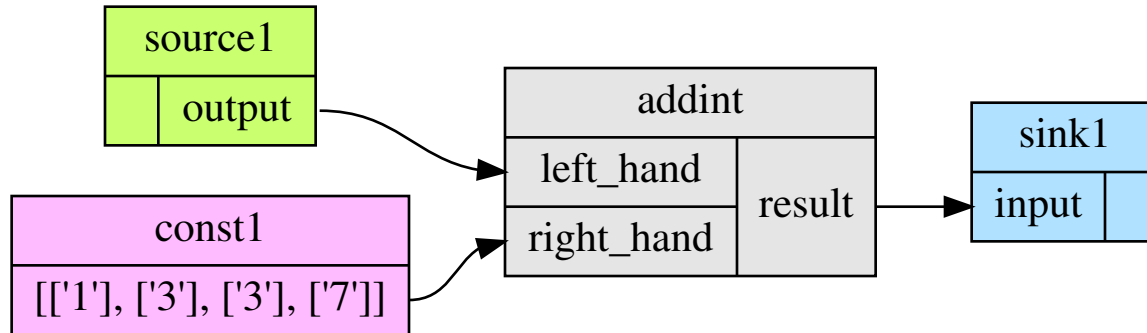
or can be create implicitly by a short hand (there are three options):

```
# This style of assignment will create a Link similar to above
>>> link = node1.outputs['image'] >> node2.inputs['image']
>>> link = node2.inputs['image'] << node1.outputs['image']
>>> node2.inputs['image'] = node1.outputs['image']
```

Note that a *Link* is also create automatically when using the short-hand for the ConstantNode `<fastr.planning.node.ConstantNode>`.

### 1.3.3 Data Flow

The data enters the Network via SourceNodes flows via other Node and leaves the Network via SinkNodes. The flow between Nodes goes from an Output via a Link to an Input. In the following image it is simple to track the data from the SourceNodes at the left to the SinkNodes at right side:



Note that the data in Fastr is stored in the Output and the Link and Input just give access to it (possible while transforming the data).

#### Data flow inside a Node

In a Node all data from the Inputs will be combined and the jobs will be generated. There are strict rules to how this combination is performed. In the default case all inputs will be used pair-wise, and if there is only a single value for an input, it will be considered as a constant.

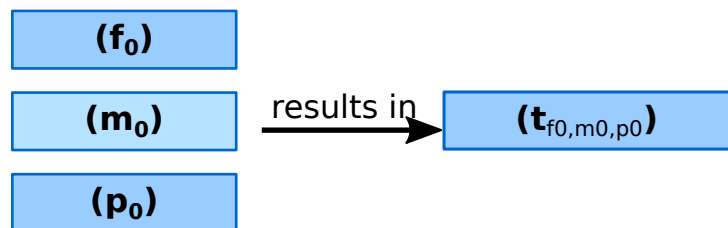
To illustrate this we will consider the following Tool (note this is a simplified version of the real tool):

```
>>> fastr.tools['Elastix']
```

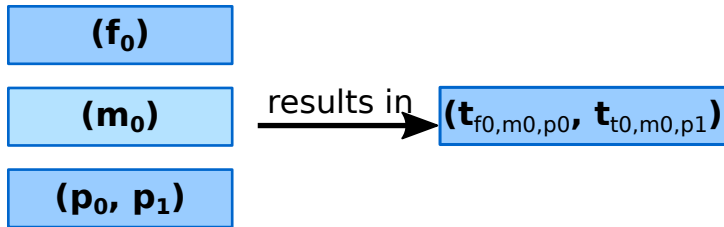
Tool Elastix v4.8 (Elastix Registration)		
Inputs		Outputs
-----		
fixed_image	(ITKImageFile)	transform_
→(ElastixTransformFile)		
moving_image	(ITKImageFile)	
parameters	(ElastixParameterFile)	

Also it is important to know that for this tool (by definition) the cardinality of the **transform** Output will match the cardinality of the **parameters** Input.

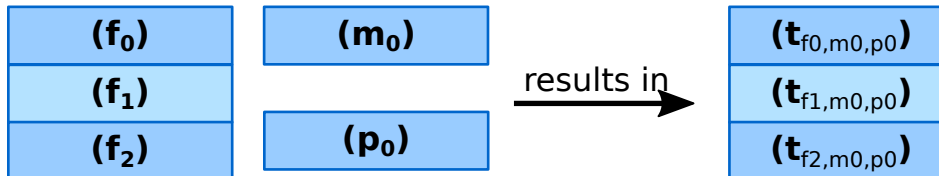
If we supply a Node based on this Tool with a single sample on each Input there will be one single matching Output sample created:



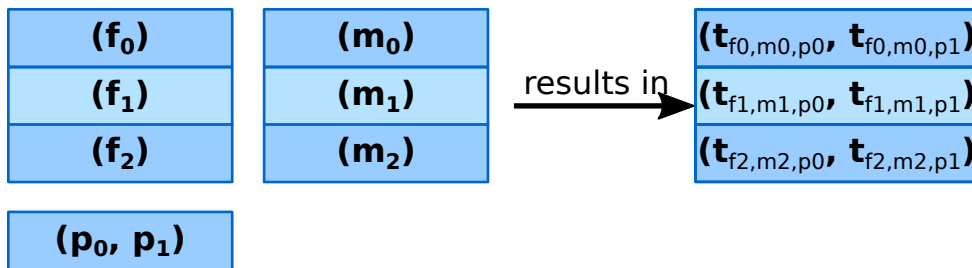
If the cardinality of the **parameters** sample would be increased to 2, the resulting **transform** sample would also become 2:



Now if the number of samples on `fixed_image` would be increased to 3, the `moving_image` and parameters will be considered constant and be repeated, resulting in 3 transform samples.



Then if the amount of samples for `moving_image` is also increased to 3, the `moving_image` and `fixed_image` will be used pairwise and the parameters will be constant.

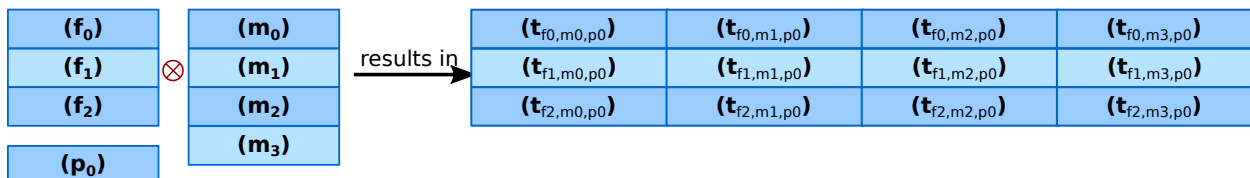


### Advanced flows in a Node

Sometimes the default pairwise behaviour is not desirable. For example if you want to test all combinations of certain input samples. To achieve this we can change the `input_group` of `Inputs` to set them apart from the rest. By default all `Inputs` are assigned to the default input group. Now let us change that:

```
>>> node = network.create_node('Elastix', id='elastix')
>>> node.inputs['moving_image'].input_group = 'moving'
```

This will result in `moving_image` to be put in a different input group. Now if we would supply `fixed_image` with 3 samples and `moving_image` with 4 samples, instead of an error we would get the following result:

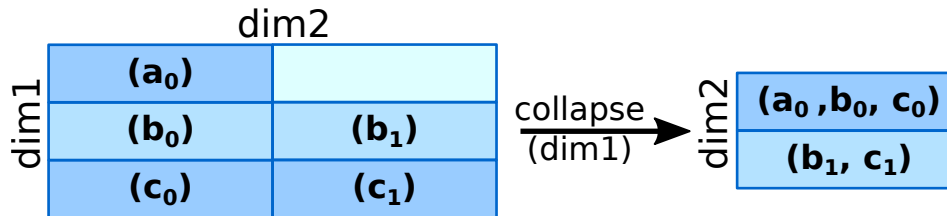


**Warning:** TODO: Expand this section with the merging dimensions

## Data flows in a Link

As mentioned before the data flows from an Output to an Input through a Link. By default the Link passed the data as is, however there are two special directives that change the shape of the data:

1. Collapsing flow, this collapses certain dimensions from the sample array into the cardinality. As a user you have to specify the dimension or tuple of dimensions you want to collapse.

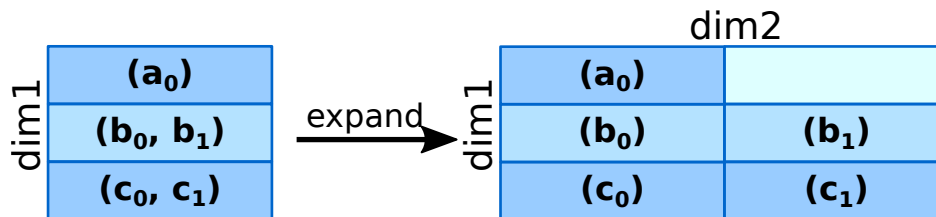


This is useful in situation where you want to use a tool that aggregates over a number of samples (e.g. take a mean or sum).

To achieve this you can set the `collapse` property of the `Link` as follows:

```
>>> link.collapse = 'dim1'
>>> link.collapse = ('dim1', 'dim2') # In case you want to collapse multiple
↳ dimensions
```

2. Expanding flow, this turns the cardinality into a new dimension. The new dimension will be named after the `Output` from which the link originates. It will be in the form of `{nodeid}__{outputid}`

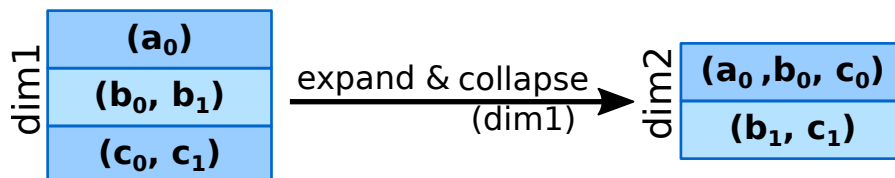


This flow directive is useful if you want to split a large sample in multiple smaller samples. This could be because processing the whole sample is not feasible because of resource constraints. An example would be splitting a 3D image into slices to process separately to avoid high memory use or to achieve parallelism.

To achieve this you can set the `expand` property of the `Link` to `True`:

```
>>> link.expand = True
```

**Note:** both collapsing and expanding can be used on the same link, it will executes similar to a expand-collapse sequence, but the newly created expand dimension is ignored in the collapse.



```
>>> link.collapse = 'dim1'
>>> link.expand = True
```

## Data flows in an Input

If an Input has multiple Links attached to it, the data will be combined by concatenating the values for each corresponding sample in the cardinality.

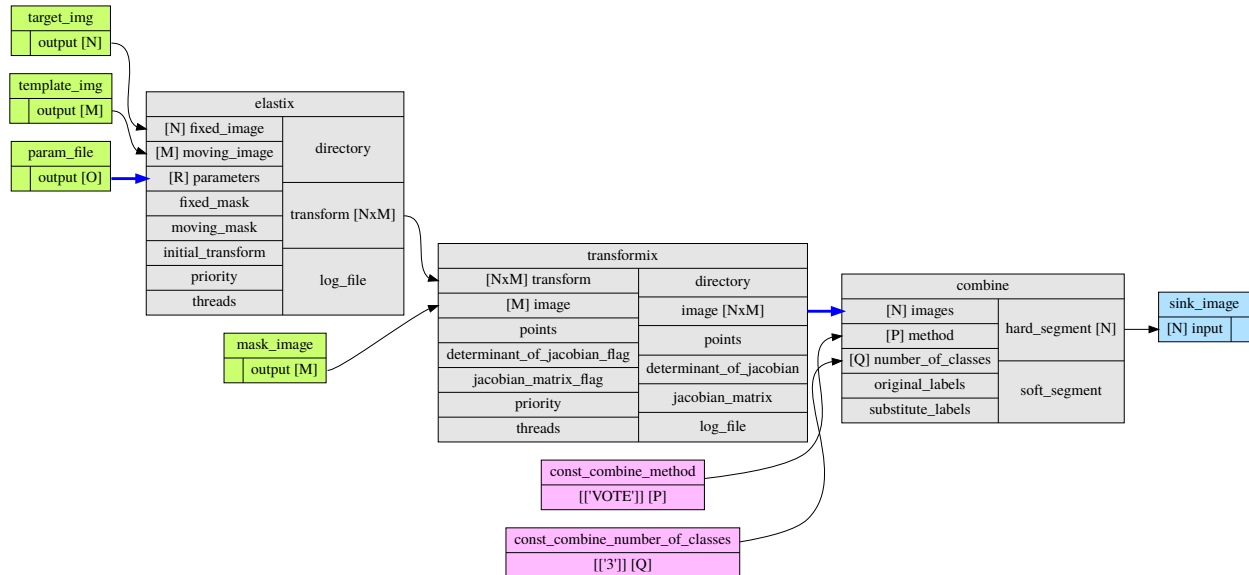
## Broadcasting (matching data of different dimensions)

Sometimes you might want to combine data that does not have the same number of dimensions. As long as all dimensions of the lower dimensional datasets match a dimension in the higher dimensional dataset, this can be achieved using *broadcasting*. The term *broadcasting* is borrowed from [NumPy](#) and described as:

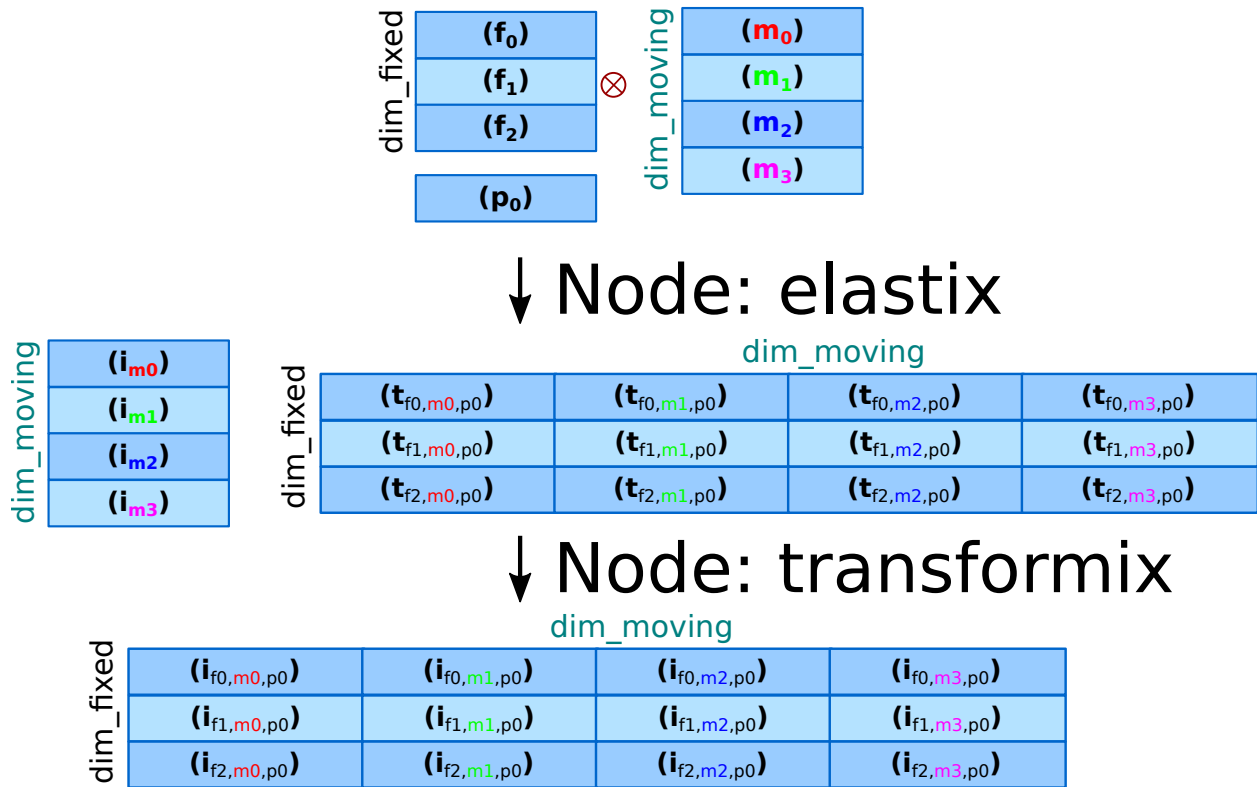
“The term broadcasting describes how numpy treats arrays with different shapes during arithmetic operations. Subject to certain constraints, the smaller array is “broadcast” across the larger array so that they have compatible shapes.”

—[NumPy manual on broadcasting](#)

In fastr it works similar, but to combined different Inputs in an InputGroup. To illustrate broadcasting it is best to use an example, the following network uses broadcasting in the `transformix` Node:



As you can see this visualization prints the dimensions for each Input and Output (e.g. the `elastix.fixed_image` Input has dimensions `[N]`). To explain what happens in more detail, we present an image illustrating the details for the samples in `elastix` and `transformix`:



In the figure the moving\_image (and references to it) are identified with different colors, so they are easy to track across the different steps.

At the top the Inputs for the `elastix` Node are illustrated. Because the input groups a set differently, output samples are generated for all combinations of `fixed_image` and `moving_image` (see *Advanced flows in a Node* for details).

In the `transformix` Node, we want to combine a list of samples that is related to the `moving_image` (it has the same dimension name and sizes) with the resulting `transform` samples from the `elastix` Node. As you can see the sizes of the sample collections do not match ( $[N]$  vs  $[N \times M]$ ). This is where *broadcasting* comes into play, it allows the system to match these related sample collections. Because all the dimensions in  $[N]$  are known in  $[N \times M]$ , it is possible to match them uniquely. This is done automatically and the result is a new  $[N \times M]$  sample collection. To create a matching sample collections, the samples in the `transformix.image` Input are reused as indicated by the colors.

**Warning:** Note that this might fail when there are data-blocks with non-unique dimension names, as it will be not be clear which of the dimensions with identical names should be matched!

### 1.3.4 DataTypes

In Fastr all data is contained in object of a specific type. The types in Fastr are represented by classes that subclass `BaseDataType`. There are a few different other classes under `BaseDataType` that are each a base class for a family of types:

- `DataType` – The base class for all types that hold data
  - `ValueType` – The base class for types that contain simple data (e.g. Int, String) that can be represented as a str
  - `EnumType` – The base class for all types that are a choice from a set of options



- *URLType* – The base class for all types that have their data stored in files (which are referenced by URL)
- *TypeGroup* – The base class for all types that actually represent a group of types

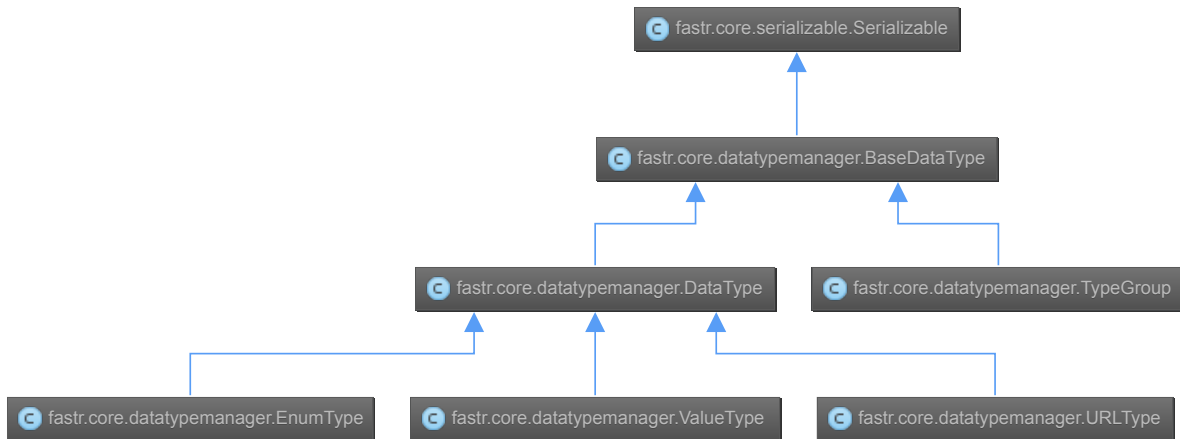


Fig. 1.2: The relation between the different DataType classes

The types are defined in xml files and created by the `DataTypeManager`. The `DataTypeManager` acts as a container containing all Fastr types. It is automatically instantiated as `fastr.types`. In fastr the created `DataType` classes are also automatically place in the `fastr.datatypes` module once created.

## Resolving Datatypes

*Outputs* in fastr can have a *TypeGroup* or a number of *DataTypes* associated with them. The final *DataType* used will depend on the linked *Inputs*. The *DataType* resolving works as a two-step procedure.

1. All possible *DataTypes* are determined and considered as *options*.
2. The best possible *DataType* from *options* is selected for non-automatic Outputs

The *options* are defined as the intersection of the set of possible values for the *Output* and each separate *Input* connected to the *Output*. Given the resulting *options* there are three scenarios:

- If there are no valid *DataTypes* (*options* is empty) the result will be None.
- If there is a single valid *DataType*, then this is automatically the result (even if it is not a preferred *DataType*).
- If there are multiple valid *DataTypes*, then the preferred *DataTypes* are used to resolve conflicts.

There are a number of places where the preferred *DataTypes* can be set, these are used in the order as given:

1. The *preferred* keyword argument to `match_types`
2. The preferred types specified in the `fastr.config`

### 1.3.5 Execution

Executing a Network is very simple:

```
>>> source_data = {'source_id1': ['val1', 'val2'],
                  'source_id2': {'id3': 'val3', 'id4': 'val4'}}
>>> sink_data = {'sink_id1': 'vfs://some_output_location/{sample_id}/file.txt'}
>>> network.execute(source_data, sink_data)
```

The `Network.execute` method takes a `dict` of source data and a `dict` sink data as arguments. The dictionaries should have a key for each `SourceNode` or `SinkNode`.

The execution of a Network uses a layered model:

- `Network.execute` will analyze the Network and call all Nodes.
- `Node.execute` will create jobs and fill their payload
- `execute_job` will execute the job on the execute machine and resolve any deferred values (`val://` urls).
- `Tool.execute` will find the correct target and call the interface and if required resolve `vfs://` urls
- `Interface.execute` will actually run the required command(s)

The `ExecutionPlugin` will call the `executionscript.py` for each job, passing the job as a gzipped pickle file. The `executionscript.py` will resolve deferred values and then call `Tool.execute` which analyses the required target and executes the underlying `Interface`. The `Interface` actually executes the job and collect the results. The result is returned (via the `Tool`) to the `executionscript.py`. There we save the result, provenance and profiling in a new gzipped pickle file. The execution system will use a callback to load the data back into the Network.

The selection and settings of the `ExecutionPlugin` are defined in the *fastr config*.

### Continuing a Network

Normally a random temporary directory is created for each run. To continue a previously stopped/crashed network, you should call the `Network.execute` method using the same temporary directory(tmp dir). You can set the temporary directory to a fixed value using the following code:

```
>>> tmpdir = '/tmp/example_network_rerun'
>>> network.execute(source_data, sink_data, tmpdir=tmpdir)
```

**Warning:** Be aware that at this moment, Fastr will rerun only the jobs where not all output files are present or if the job/tool parameters have been changed. It will not rerun if the input data of the node has changed or the actual tools have been adjusted. In these cases you should remove the output files of these nodes, to force a rerun.

### 1.3.6 IOPlugins

Sources and sink are used to get data in and out of a `Network` during execution. To make the data retrieval and storage easier, a plugin system was created that selects different plugins based on the URL scheme used. So for example, a url starting with `vfs://` will be handles by the *VirtualFileSystem plugin*. A list of all the *IOPlugins* known by the system and their use can be found at *IOPlugin Reference*.

### 1.3.7 Secrets

Fastr uses a secrets system for storing and retrieving login credentials. Currently the following keyrings are supported:

- Python keyring and keyrings.alt lib: - Mac OS X Keychain - Freedesktop Secret Service (requires secretstorage) - KWallet (requires dbus) - Windows Credential Vault - Gnome Keyring - Google Keyring (stores keyring on Google Docs) - Windows Crypto API (File-based keyring secured by Windows Crypto API) - Windows Registry Keyring (registry-based keyring secured by Windows Crypto API) - PyCrypto File Keyring - Plaintext File Keyring (not recommended)
- Netrc (not recommended)

When a password is retrieved through the fastr SecretService it loops through all of the available SecretProviders (currently keyring and netrc) until a match is found.

The Python keyring library automatically picks the best available keyring backend. If you wish to choose your own python keyring backend it is possible to do so by make a keyring configuration file according to the keyring library documentation. The python keyring library connects to one keyring. Currently it cannot loop through all available keyrings until a match is found.

### 1.3.8 Debugging

This section is about debugging Fastr tools wrappers, Fastr Networks (when building a Network) and Fastr Network Runs.

#### Debugging a Fastr tool

When wrapping a Tool in Fastr sometimes it will not work as expected or not load properly. Fastr is shipped with a command that helps checking Tools. The *fastr verify* command can try to load a Tool in steps to make it more easy to understand where the loading went wrong.

The *fastr verify* command will use the following steps:

- Try to load the tool with and without compression
- Try to find the correct serializer and make sure the format is correct
- Try to validate the Tool content against the json\_schema of a proper Tool
- Try to create a Tool object
- If available, execute the tool test

An example of the use of *fastr verify*:

```
$ fastr verify tool fastr/resources/tools/fastr/math/0.1/add.xml
[INFO]    verify:0020 >> Trying to read file with compression OFF
[INFO]    verify:0036 >> Read data from file successfully
[INFO]    verify:0040 >> Trying to load file using serializer "xml"
[INFO]    verify:0070 >> Validating data against Tool schema
[INFO]    verify:0080 >> Instantiating Tool object
[INFO]    verify:0088 >> Loaded tool <Tool: Add version: 1.0> successfully
[INFO]    verify:0090 >> Testing tool...
```

If your Tool is loading but not functioning as expected you might want to easily test your Tool without building an entire Network around it that can obscure errors. It is possible to run a tool from the Python prompt directly using *tool.execute*:

```
>>> tool.execute(left_hand=40, right_hand=2)
[INFO] localbinarytarget:0090 >> Changing ./bin
[INFO] tool:0311 >> Target is <Plugin: LocalBinaryTarget>
[INFO] tool:0318 >> Using payload: {'inputs': {'right_hand': (2,), 'left_hand': (40,
↪)}, 'outputs': {}}
[INFO] localbinarytarget:0135 >> Adding extra PATH: ['/home/hachterberg/dev/fastr-
↪develop/fastr/fastr/resources/tools/fastr/math/0.1/bin']
[INFO] fastrinterface:0393 >> Execution payload: {'inputs': {'right_hand': (2,), 'left_
↪hand': (40,)}, 'outputs': {}}
[INFO] fastrinterface:0496 >> Adding (40,) to argument list based on <fastrinterface.
↪InputParameterDescription object at 0x7fc950fa8850>
[INFO] fastrinterface:0496 >> Adding (2,) to argument list based on <fastrinterface.
↪InputParameterDescription object at 0x7fc950fa87d0>
[INFO] localbinarytarget:0287 >> Options: ['/home/hachterberg/dev/fastr-develop/fastr/
↪fastr/resources/tools/fastr/math/0.1/bin']
[INFO] localbinarytarget:0201 >> Calling command arguments: ['python', '/home/
↪hachterberg/dev/fastr-develop/fastr/fastr/resources/tools/fastr/math/0.1/bin/addint.py
↪', '--in1', '40', '--in2', '2']
[INFO] localbinarytarget:0205 >> Calling command: "'python' '/home/hachterberg/dev/fastr-
↪develop/fastr/fastr/resources/tools/fastr/math/0.1/bin/addint.py' '--in1' '40' '--in2'
↪'2'"
[INFO] fastrinterface:0400 >> Collecting results
[INFO] executionpluginmanager:0467 >> Callback processing thread ended!
[INFO] executionpluginmanager:0467 >> Callback processing thread ended!
[INFO] executionpluginmanager:0467 >> Callback processing thread ended!
[INFO] jsoncollector:0076 >> Setting data for result with [42]
<fastr.core.interface.InterfaceResult at 0x7fc9661ccfd0>
```

In this case an AddInt was ran from the python shell. As you can see it shows the payload it created based on the call, followed by the options for the directories that contain the binary. Then the command that is called is given both as a list and string (for easy copying to the prompt yourself). Finally the collected results is displayed.

---

**Note:** You can give input and outputs as keyword arguments for execute. If an input and output have the same name, you can disambiguate them by prefixing them with `in_` or `out_` (e.g. `in_image` and `out_image`)

---

## Debugging an invalid Network

The simplest command to check if your Network is considered valid is to use the `Network.is_valid` method. It will simply check if the Network is valid:

```
>>> network.is_valid()
True
```

It will return a boolean that only indicates the validity of the Network, but it will print any errors it found to the console/log with the ERROR log level, for example when datatypes on a link do not match:

```
>>> invalid_network.is_valid()
[WARNING] datatypeemanager:0388 >> No matching DataType available (args (<ValueType:
↪Float class [Loaded]>, <ValueType: Int class [Loaded]>))
[WARNING] link:0546 >> Cannot match datatypes <ValueType: Float class [Loaded]> and
↪<ValueType: Int class [Loaded]> or not preferred datatype is set! Abort linking fastr:/
```

(continues on next page)

(continued from previous page)

```

↪//networks/add_ints/0.0/nodelist/source/outputs/output to fastr:///networks/add_ints/0.
↪0/nodelist/add/inputs/left_hand!
[WARNING] datatypepmanager:0388 >> No matching DataType available (args (<ValueType:
↪Float class [Loaded]>, <ValueType: Int class [Loaded]>))
[ERROR] network:0571 >> [add] Input left_hand is not valid: SubInput fastr:///networks/
↪add_ints/0.0/nodelist/add/inputs/left_hand/0 is not valid: SubInput source (link_0) is
↪not valid
[ERROR] network:0571 >> [add] Input left_hand is not valid: SubInput fastr:///networks/
↪add_ints/0.0/nodelist/add/inputs/left_hand/0 is not valid: [link_0] source and target
↪have non-matching datatypes: source Float and Int
[ERROR] network:0571 >> [link_0] source and target have non-matching datatypes: source
↪Float and Int
False

```

Because the messages might not always be enough to understand errors in the more complex Networks, we would advice you to create a plot of the network using the `network.draw_network` method:

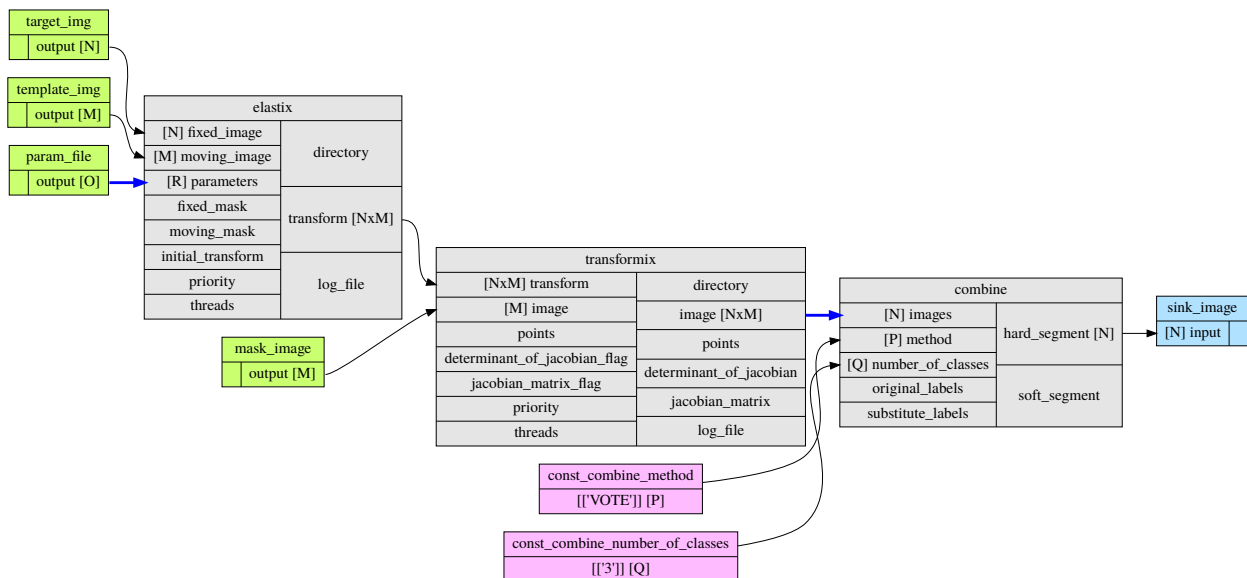
```

>>> network.draw_network(network.id, draw_dimensions=True, expand_macro=True)
'add_ints.svg'

```

The value returned is the path of the output image generated (it will be placed in the current working directory). The `draw_dimensions=True` will make the drawing add indications about the sample dimensions in each Input and Output, whereas `expand_macro=True` causes the draw to expand MacroNodes and draw the content of them. If you have many nested MacroNodes, you can set `expand_macro` to an integer and that is the depth until which the MacroNodes will be draw in detail.

An example of a simple multi-atlas segmentation Network nicely shows the use of drawing the dimensions, the dimensions vary in certain Nodes due to the use of input\_groups and a collapsing link (drawn in blue):



## Debugging a Network run with errors

If a Network run did finish but there were errors detected, Fastr will report those at the end of the execution. We included an example of a Network that has failing samples in `fastr/examples/failing_network.py` which can be used to test debugging. An example of the output of a Network run with failures:

```
[INFO] networkrun:0604 >> #####
[INFO] networkrun:0605 >> #    network execution FINISHED    #
[INFO] networkrun:0606 >> #####
[INFO] networkrun:0618 >> ===== RESULTS =====
[INFO] networkrun:0627 >> sink_1: 2 success / 2 failed
[INFO] networkrun:0627 >> sink_2: 2 success / 2 failed
[INFO] networkrun:0627 >> sink_3: 1 success / 3 failed
[INFO] networkrun:0627 >> sink_4: 1 success / 3 failed
[INFO] networkrun:0627 >> sink_5: 1 success / 3 failed
[INFO] networkrun:0628 >> =====
[WARNING] networkrun:0651 >> There were failed samples in the run, to start debugging
↪you can run:

    fastr trace $RUNDIR/__sink_data__.json --sinks

see the debug section in the manual at https://fastr.readthedocs.io/en/default/static/
↪user_manual.html#debugging for more information.
```

As you can see, there were failed samples in every sink. Also you already get the suggestion to use `fastr trace`. This command helps you inspect the staging directory of the Network run and pinpoint the errors.

The suggested command will print a similar summary as given by the network execution:

```
$ fastr trace $RUNDIR/__sink_data__.json --sinks
sink_1 -- 2 failed -- 2 succeeded
sink_2 -- 2 failed -- 2 succeeded
sink_3 -- 3 failed -- 1 succeeded
sink_4 -- 3 failed -- 1 succeeded
sink_5 -- 3 failed -- 1 succeeded
```

Since this is not given us new information we can add the `-v` flag for more output and limit the output to one sink, in this case `sink_5`:

```
$ fastr trace $RUNDIR/__sink_data__.json --sinks sink_5
sink_5 -- 3 failed -- 1 succeeded
  sample_1_1: Encountered error: [FastrOutputValidationError] Could not find result for
↪output out_2 (/home/hachterberg/dev/fastr-develop/fastr/fastr/execution/job.py:970)
  sample_1_2: Encountered error: [FastrOutputValidationError] Could not find result for
↪output out_1 (/home/hachterberg/dev/fastr-develop/fastr/fastr/execution/job.py:970)
  sample_1_3: Encountered error: [FastrOutputValidationError] Could not find result for
↪output out_1 (/home/hachterberg/dev/fastr-develop/fastr/fastr/execution/job.py:970)
  sample_1_3: Encountered error: [FastrOutputValidationError] Could not find result for
↪output out_2 (/home/hachterberg/dev/fastr-develop/fastr/fastr/execution/job.py:970)
```

Now we are given one error per sample, but this does not yet give us that much information. To get a very detailed report we have to specify one sink and one sample. This will make the `fastr trace` command print a complete error report for that sample:

```

$ fastr trace $RUNDIR/__sink_data__.json --sinks sink_5 --sample sample_1_1 -v
Tracing errors for sample sample_1_1 from sink sink_5
Located result pickle: /home/hachterberg/FastrTemp/fastr_failing_network_2017-09-04T10-
↳44-58_uMWeMV/step_1/sample_1_1/__fastr_result__.pickle.gz

===== JOB failing_network__step_1__sample_1_1 =====
Network: failing_network
Run: failing_network_2017-09-04T10-44-58
Node: step_1
Sample index: (1)
Sample id: sample_1_1
Status: JobState.execution_failed
Timestamp: 2017-09-04 08:45:19.238192
Job file: /home/hachterberg/FastrTemp/fastr_failing_network_2017-09-04T10-44-58_uMWeMV/
↳step_1/sample_1_1/__fastr_result__.pickle.gz

Command:
List representation: [u'python', u'/home/hachterberg/dev/fastr-develop/fastr/fastr/
↳resources/tools/fastr/util/0.1/bin/fail.py', u'--in_1', u'1', u'--in_2', u'1', u'--
↳fail_2']
String representation: 'python' '/home/hachterberg/dev/fastr-develop/fastr/fastr/
↳resources/tools/fastr/util/0.1/bin/fail.py' '--in_1' '1' '--in_2' '1' '--fail_2'

Output data:
{'out_1': [<Int: 2>]}

Status history:
2017-09-04 08:45:19.238212: JobState.created
2017-09-04 08:45:21.537417: JobState.running
2017-09-04 08:45:31.578864: JobState.execution_failed

----- ERRORS -----
- FastrOutputValidationError: Could not find result for output out_2 (/home/hachterberg/
↳dev/fastr-develop/fastr/fastr/execution/job.py:970)
- FastrValueError: [failing_network__step_1__sample_1_1] Output values are not valid!
↳(/home/hachterberg/dev/fastr-develop/fastr/fastr/execution/job.py:747)

-----

----- STDOUT -----
Namespace(fail_1=False, fail_2=True, in_1=1, in_2=1)
in 1 : 1
in 2 : 1
fail_1: False
fail_2: True
RESULT_1=[2]

-----

----- STDERR -----

-----

```

As shown above, it finds the result files of the failed job(s) and prints the most important information. The first paragraph shows the information about the Job that was involved. The second paragraph shows the command used both as a list (which is clearer and internally used in Python) and as a string (which you can copy/paste to the shell to test the command). Then there is the output data as determined by Fastr. The next section shows the status history of the Job which can give an indication about wait and run times. Then there are the errors that Fastr encountered during the execution of the Job. In this case it could not find the output for the Tool. Finally the stdout and stderr of the subprocess are printed. In this case we can see that `RESULT_2=[...]` was not in the stdout, and so the result could not be located.

---

**Note:** Sometimes there are no Job results in a directory, this usually means the process got killed before the Job could finished. On cluster environments, this often means that the process was killed due to memory constraints.

---

## Asking for help with debugging

If you would like help with debugging, you can contact us via the [fastr-users google group](#). To enable us to track the errors please include the following:

- The entire log of the fastr run (can be copied from console or from the end of `~/ .fastr/logs/info.log`).
- A dump of the network run, which can be created that by using the *fastr dump* command like:

```
$ fastr dump $RUNDIR fastr_run_dump.zip
```

This will create a zip file including all the job files, logs, etc but not the actual data files.

These should be enough information to trace most errors. In some cases we might need to ask for additional information (e.g. tool files, datatype files) or actions from your side.

## 1.3.9 Naming Convention

For the naming convention of the tools we tried to stay close to the Python [PEP 8](#) coding style. In short, we defined toolnames as classes so they should be UpperCamelCased. The inputs and outputs of a tool we considered as functions or method arguments, these should we named lower\_case\_with\_underscores.

An overview of the mapping of Fastr to [PEP 8](#):

Fastr construct	Python <a href="#">PEP8</a> equivalent	Examples
Network.id	<a href="#">module</a>	brain_tissue_segmentation
Tool.id	<a href="#">class</a>	BrainExtractionTool, ThresholdImage
Node.id	<a href="#">variable name</a>	brain_extraction, threshold_mask
Input/Output.id	<a href="#">method</a>	image, number_of_classes, probability_image

Furthermore there are some small guidelines:

- No input or output in the input or output names. This is already specified when setting or getting the data.
- Add the type of the output that is named. i.e. enum, string, flag, image,
  - No File in the input/output name (Passing files around is what Fastr was developed for).
  - No type necessary where type is implied i.e. lower\_threshold, number\_of\_levels, max\_threads.
- Where possible/useful use the fullname instead of an abbreviation.



### 1.3.10 Provenance

For every data derived data object, Fastr records the **Provenance**. The *SinkNode* write provenance records next to every data object it writes out. The records contain information on what operations were performed to obtain the resulting data object.

#### W3C Prov

The provenance is recorded using the **W3C Prov Data Model (PROV-DM)**. Behind the scenes we are using the python `prov` implementation.

The PROV-DM defines 3 Starting Point Classes and their relating properties. See [Fig. 1.3](#) for a graphic representation of the classes and the relations.

Fig. 1.3: The three Starting Point classes and the properties that relate them. The diagrams in this document depict Entities as yellow ovals, Activities as blue rectangles, and Agents as orange pentagons. The responsibility properties are shown in pink.<sup>0</sup>

\*

#### Implementation

In the workflow document the provenance classes map to fastr concepts in the following way:

**Agent**  
Fastr, *Networks*, *Tools*, *Nodes*

**Activity**  
*Jobs*

**Entities**  
Data

#### Usage

The provenance is stored in `ProvDocument` objects in pickles. The convenience command line tool `fastr prov` can be used to extract the provenance in the **PROV-N** notation and can be serialized to **PROV-JSON** and **PROV-XML**. The provenance document can also be vizualized using the `fastr prov` command line tool.

## 1.4 Command Line Tools

Fastr is shipped with a number of command line tools to perform common tasks and greatly simplify things such as debugging. The list of command line tools that is included in Fastr:

<sup>0</sup> This picture and caption is taken from <http://www.w3.org/TR/prov-o/>. “Copyright © 2011-2013 World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang). <http://www.w3.org/Consortium/Legal/2015/doc-license>”

command	description
<i>cat</i>	Print information from a job file
<i>dump</i>	Dump the contents of a network run tempdir into a zip for remote assistance
<i>execute</i>	Execute a fastr job file
<i>extract_argparse</i>	Create a stub for a Tool based on a python script using argparse
<i>provenance</i>	Get PROV information from the result pickle.
<i>pylint</i>	Tiny wrapper in pylint so the output can be saved to a file (for test automation)
<i>report</i>	Print report of a job result (__fastr_result__.pickle.gz) file
<i>run</i>	Run a Network from the commandline
<i>sink</i>	Command line access to the IOPlugin sink
<i>source</i>	Command line access to the IOPlugin source
<i>test</i>	Run the tests of a tool to verify the proper function
<i>trace</i>	Trace samples/sinks from a run
<i>upgrade</i>	Upgrade a fastr 2.x python file to fastr 3.x syntax
<i>verify</i>	Verify fastr resources, at the moment only tool definitions are supported.

### 1.4.1 fastr cat

Extract selected information from the extra job info. The path is the selection of the data to retrieve. Every parts of the path (separated by a /) is seen as the index for the previous object. So for example to get the stdout of a job, you could use 'fastr cat \_\_fastr\_extra\_job\_info\_\_.json process/stdout'.

```
usage: fastr cat [-h] __fastr_extra_job_info__.json path
```

#### Positional Arguments

**\_\_fastr\_extra\_job\_info\_\_.json** result file to cat

**path** path of the data to print

### 1.4.2 fastr dump

Create a dump of a network run directory that contains the most important information for debugging. This includes a serialization of the network, all the job command and result files, the extra job information files and the provenance files. No data files will be included, but note that if jobs get sensitive information passed via the command line this will be included in the job files.

```
usage: fastr dump [-h] RUNDIR DUMP.zip
```

### Positional Arguments

<b>RUNDIR</b>	The run directory to dump
<b>DUMP.zip</b>	The file to place the dump in

### 1.4.3 fastr execute

Execute a job from commandline.

```
usage: fastr execute [-h] [JOBFILE]
```

### Positional Arguments

<b>JOBFILE</b>	File of the job to execute (default ./__fastr_command__.yaml)
----------------	---

### 1.4.4 fastr extract\_argparse

Extract basic information from argparse.

```
usage: fastr extract_argparse [-h] SCRIPT.py TOOL.xml
```

### Positional Arguments

<b>SCRIPT.py</b>	Python script to inspect
<b>TOOL.xml</b>	created Tool stub

### 1.4.5 fastr provenance

Export the provenance information from JSON to other formats or plot the provenance data as a graph.

```
usage: fastr provenance [-h] [-so SYNTAX_OUT_FILE] [-sf SYNTAX_FORMAT]
                        [-i INDENT] [-vo VISUALIZE_OUT_FILE]
                        [RESULTFILE]
```

## Positional Arguments

**RESULTFILE**      File of the job to execute (default `./__fastr_prov__.json`)

## Named Arguments

**-so, --syntax-out-file** Write the syntax to file.  
**-sf, --syntax-format** Choices are: [json], provn or xml  
Default: “json”  
**-i, --indent** Indent size of the serialized documents.  
Default: 2  
**-vo, --visualize-out-file** Visualize the provenance. The most preferred format is svg. You can specify any format pydot supports. Specify the format by postfixing the filename with an extension.

### 1.4.6 fastr pylint

Run pylint in such a way that the output is written to a file

```
usage: fastr pylint [-h] --output_file PYLINT.OUT
```

## Named Arguments

**--output\_file**      The file to result in

### 1.4.7 fastr report

Print a report of a job result file.

```
usage: fastr report [-h] [-v] [JOBFILE]
```

## Positional Arguments

**JOBFILE**      File of the job to execute (default `./__fastr_result__.yaml`)

## Named Arguments

**-v, --verbose** More verbose (e.g. add fastr job stdout and stderr)  
Default: False

### 1.4.8 fastr run

Execute a job or network from commandline.

```
usage: fastr run [-h] NETWORKFILE
```

## Positional Arguments

**NETWORKFILE** File of the network to execute

### 1.4.9 fastr sink

executes an ioplugin

```
usage: fastr sink [-h] -i INPUT [INPUT ...] -o OUTPUT [OUTPUT ...]
                [-d DATATYPE [DATATYPE ...]]
```

## Named Arguments

**-i, --input** The url to process (can also be a list)  
**-o, --output** The output urls in vfs scheme (can also be a list and should be the same size as -inurl)  
**-d, --datatype** The datatype of the source/sink data to handle

### 1.4.10 fastr source

Executes an source command

```
usage: fastr source [-h] -i INPUT [INPUT ...] -o OUTPUT [-d DATATYPE]
                  [-s SAMPLE_ID]
```

## Named Arguments

<b>-i, --input</b>	The url to process (can also be a list)
<b>-o, --output</b>	The output url in vfs scheme
<b>-d, --datatype</b>	The datatype of the source/sink data to handle
<b>-s, --sample_id</b>	The sample_id of the source/sink data to handle

### 1.4.11 fastr test

Run a tests for a fastr resource.

```
usage: fastr test [-h] {tool,tools,network,networks} ...
```

## Sub-commands

### tool

Test a single tool

```
fastr test tool [-h] TOOL
```

## Positional Arguments

<b>TOOL</b>	Tool to test or directory with tool reference data
-------------	--

### tools

Test all tools known to fastr

```
fastr test tools [-h]
```

### network

Test a single network

```
fastr test network [-h] NETWORK
```

## Positional Arguments

**NETWORK**            The reference data to test the Network

### networks

Test all network references inside subdirectories

```
fastr test networks [-h] [--result RESULT.json] REFERENCE
```

## Positional Arguments

**REFERENCE**            path of the directory containing subdirectories with reference data

## Named Arguments

**--result**            Write the results of the test to a JSON file

## 1.4.12 fastr trace

Fastr trace helps you inspect the staging directory of the Network run and pinpoint the errors.

```
usage: fastr trace [-h] [--verbose] [--sinks [SINKS [SINKS ...]]]
                  [--samples [SAMPLES [SAMPLES ...]]]
                  [--sink_data__json]
```

## Positional Arguments

**\_\_sink\_data\_\_.json**    result file to cat  
Default: “/home/docs/checkouts/readthedocs.org/user\_builds/fastr/checkouts/3.3.1/fastr/doc/\_\_sink\_data\_\_”

## Named Arguments

**--verbose, -v**        set verbose output for more details  
Default: False

**--sinks**            list results for specified sinks

**--samples**          list result for all samples

### 1.4.13 fastr upgrade

Upgrades a python file that creates a Network to the new fastr 3.x syntax. The file will be parsed and the full syntax tree will be transformed to fit the new syntax.

---

**Note:** Solves most common problems, but cannot always solve 100% of the issues

---

```
usage: fastr upgrade [-h] [--type TYPE] NETWORK.py NEW.py
```

#### Positional Arguments

<b>NETWORK.py</b>	Network creation file (in python) to upgrade
<b>NEW.py</b>	location of the result file

#### Named Arguments

<b>--type</b>	tool of resource to upgrade, one of: network, tool
---------------	--

### 1.4.14 fastr verify

Verify fastr resources, at the moment only tool definitions are supported.

```
usage: fastr verify [-h] [--createtest] TYPE path
```

#### Positional Arguments

<b>TYPE</b>	Possible choices: tool Type of resource to verify (e.g. tool)
<b>path</b>	path of the resource to verify

#### Named Arguments

<b>--createtest, -c</b>	Create a reference result for a tool test Default: False
-------------------------	---



## 1.5 Resource File Formats

This chapter describes the various files fastr uses. The function and format of the files is described allowing the user to configure fastr and add DataTypes and Tools.

### 1.5.1 Config file

Fastr reads the config files from `$FASTRHOME/config.py` by default. If the `$FASTRHOME` environment variable is not set it will default to `~/.fastr`. As a result it read:

- `$FASTRHOME/config.py` (if environment variable set)
- `~/.fastr/config.py` (otherwise)

Reading a new config file change or override settings, making the last config file read have the highest priority. All settings have a default value, making config files and all settings within optional.

---

**Note:** To verify which config files have been read you can see `fastr.config.read_config_files` which contains a list of the read config files (in read order).

---

---

**Note:** If `$FASTRHOME` is set, `$FASTRHOME/tools` is automatically added as a tool directory if it exists and `$FASTRHOME/datatypes` is automatically added as a type directory if it exists.

---

### Splitting up config files

Sometimes it is nice to have config files split in multiple smaller files. Next to the `config.py` you can also created a directory `config.d` and all `.py` files in this directory will be sourced in alphabetical order.

Given the following layout of the `$FASTRHOME` directory:

```
./config.d/a.py
./config.d/b.txt
./config.d/c.py
./config.py
```

The following files will be read in order:

1. `./config.py`
2. `./config.d/a.py`
3. `./config.d/c.py`

## Example config file

Here is a minimal config file:

```
# Enable debugging output
debug = False

# Define the path to the tool definitions
tools_path = ['/path/to/tools',
              '/path/to/other/tools'] + tools_path
types_path = ['/path/to/datatypes',
              '/path/to/other/datatypes'] + types_path

# Specify what your preferred output types are.
preferred_types += ["NiftiImageFileCompressed",
                   "NiftiImageFile"]

# Set the tmp mount
mounts['tmp'] = '/path/to/tmpdir'
```

## Format

The config file is actually a python source file. The next syntax applies to setting configuration values:

```
# Simple values
float_value = 1.0
int_value = 1
str_value = "Some value"
other_str_value = 'name'.capitalize()

# List-like values
list_value = ['over', 'ride', 'values']
other_list_value.prepend('first')
other_list_value.append('list')

# Dict-like values
dict_value = {'this': 1, 'is': 2, 'fixed': 3}
other_dict_value['added'] = 'this key'
```

---

**Note:** Dictionaries and list always have a default, so you can always append or assign elements to them and do not have to create them in a config file. Best practice is to only edit them unless you really want to block out the earlier config files.

---

Most operations will be assigning values, but for list and dict values a special wrapper object is used that allows manipulations from the default. This limits the operations allowed.

List values in the config.py have the following supported operators/methods:

- +, \_\_add\_\_ and \_\_radd\_\_
- += or \_\_iadd\_\_
- append

- `prepend`
- `extend`

Mapping (dict-like) values in the `config.py` have the following supported operators/methods:

- `update`
- `[]` or `__getitem__`, `__setitem__` and `__delitem__`

### Configuration fields

This is a table the known config fields on the system:

name	type	description	default
debug	bool	Flag to enable/disable debugging	False
examplesdir	str	Directory containing the fastr examples	\$systemdir/examples
execution_plugin	str	The default execution plugin to use	'ProcessPoolExecution'
execution-script	str	Execution script location	\$systemdir/execution/executionscript.py
extra_config_dirs	list	Extra configuration directories to read	['']
filesynchelpstr_url	str	Redis url e.g. redis://localhost:6379	''
job_cleanup_level	int	The level of cleanup required, options: all, no_cleanup, non_failed	no_cleanup
log_to_file	bool	Indicate if default logging settings should log to files or not	False
logdir	str	Directory where the fastr logs will be placed	\$userdir/logs
logging_config	dict	Python logger config	{}
loglevel	int	The log level to use (as int), INFO is 20, WARNING is 30, etc	20
logtype	str	Type of logging to use	'default'
mounts	dict	A dictionary containing all mount points in the VFS system	{'tmp': '\$TMPDIR', 'examples': '\$systemdir/examples', 'example_data': '\$systemdir/examples/data', 'home': '~/', 'fastr_home': '\$FASTRHOME or ~/.fastr'}
networks_path	list	Directories to scan for networks	['\$userdir/networks', '\$resourcedir/networks']
plugins_path	list	Directories to scan for plugins	['\$userdir/plugins', '\$resourcedir/plugins']
preferred_types	list	A list indicating the order of the preferred types to use. First item is most preferred.	[]
protected_modules	list	A list of modules in the environnet modules that are protected against unloading	[]
queue_report_interval	int	Interval in which to report the number of queued jobs (default is 0, no reporting)	0
reporting_plugins	list	The reporting plugins to use, is a list of all plugins to be activated	['SimpleReport']
resourcesdir	str	Directory containing the fastr system resources	\$systemdir/resources
schemadir	str	Directory containing the fastr data schemas	\$systemdir/schemas
source_job_limit	int	The number of source jobs allowed to run concurrently	0
systemdir	str	Fastr installation directory	Directory of the top-level fastr package
tools_path	list	Directories to scan for tools	['\$userdir/tools', '\$resourcedir/tools']
types_path	list	Directories to scan for datatypes	['\$userdir/datatypes', '\$resourcedir/datatypes']
userdir	str	Fastr user configuration directory	\$FASTRHOME or ~/.fastr
warn_devel	bool	Warning users on import if this	True

**Note:** This table only includes the fastr default config fields, but not the fields added by plugins. For information look at the appropriate plugin reference. For the built-in fastr plugins they can be found at the [plugin reference](#)

## 1.5.2 Tool description

*Tools* are the building blocks in the fastr network. To add new *Tools* to fastr, XML/json files containing a *Tool* definition can be added. These files have the following layout:

Attribute		Description	
id		The id of this Tool (used internally in fastr)	
name		The name of the Tool, for human readability	
version		The version of the Tool wrapper (not the binary)	
url		The url of the Tool wrapper	
authors[]		List of authors of the Tools wrapper	
	name	Name of the author	
	email	Email address of the author	
	url	URL of the website of the author	
tags	tag[]	List of tags describing the Tool	
command		Description of the underlying command	
	version	Version of the tool that is wrapped	
	url	Website where the tools that is wrapped can be obtained	
	targets[]		Description of the target binaries/script of this Tool
		os	OS targeted (windows, linux, macos or * (for any)
		arch	Architecture targeted 32, 64 or * (for any)
		...	Extra variables based on the target used, see <i>Targets</i>
	description	Description of the Tool	
	license	License of the Tool, either full license or a clear name (e.g. LGPL, GPL v2)	
	authors[]		List of authors of the Tool (not the wrapper!)
		name	Name of the authors
		email	Email address of the author
		url	URL of the website of the author
interface		The interface definition see <i>Interfaces</i>	
help		Help text explaining the use of the Tool	
cite		Bibtext of the Citation(s) to reference when using this Tool for a publication	

## 1.6 Plugin Reference

In this chapter we describe the different plugins bundled with Fastr (e.g. IOPlugins, ExecutionPlugins). The reference is build automatically from code, so after installing a new plugin the documentation has to be rebuild for it to be included in the docs.

### 1.6.1 CollectorPlugin Reference

CollectorPlugins are used for finding and collecting the output data of outputs part of a `FastrInterface`

scheme	CollectorPlugin
JsonCollector	<i>JsonCollector</i>
PathCollector	<i>PathCollector</i>
StdoutCollector	<i>StdoutCollector</i>

#### JsonCollector

The JsonCollector plugin allows a program to print out the result in a pre-defined JSON format. It is then used as values for fastr.

The working is as follows:

1. The location of the output is taken
2. If the location is `None`, go to step 5
3. The substitutions are performed on the location field (see below)
4. The location is used as a [regular expression](#) and matched to the stdout line by line
5. The matched string (or entire stdout if location is `None`) is [loaded as a json](#)
6. The data is parsed by `set_result`

The structure of the JSON has to follow the a predefined format. For normal Nodes the format is in the form:

```
[value1, value2, value3]
```

where the multiple values represent the cardinality.

For a `FlowNodes` the format is the form:

```
{
  'sample_id1': [value1, value2, value3],
  'sample_id2': [value4, value5, value6]
}
```

This allows the tool to create multiple output samples in a single run.

## PathCollector

The PathCollector plugin for the FastrInterface. This plugin uses the location fields to find data on the filesystem. To use this plugin the method of the output has to be set to path

The general working is as follows:

1. The location field is taken from the output
2. The substitutions are performed on the location field (see below)
3. The updated location field will be used as a [regular expression](#) filter
4. The filesystem is scanned for all matching files/directory

The special substitutions performed on the location use the Format Specification Mini-Language [Format Specification Mini-Language](#). The predefined fields that can be used are:

- `inputs`, an object with the input values (use like `{inputs.image[0]}`) The input contains the following attributes that you can access:
  - `.directory` for the directory name (use like `input.image[0].directory`) The directory is the same as the result of `os.path.dirname`
  - `.filename` is the result of `os.path.basename` on the path
  - `.basename` for the basename name (use like `input.image[0].basename`) The basename is the same as the result of `os.path.basename` and the extension stripped. The extension is considered to be everything after the first dot in the filename.
  - `.extension` for the extension name (use like `input.image[0].extension`)
- `output`, an object with the output values (use like `{outputs.result[0]}`) It contains the same attributes as the input
  - `special.cardinality`, the index of the current cardinality
  - `special.extension`, is the extension for the output DataType

Example use:

```
<output ... method="path" location="{output.directory[0]}/TransformParameters.{special.
↪cardinality}.{special.extension}"/>
```

Given the output directory `./nodeid/sampleid/result`, the second sample in the output and filetype with a `txt` extension, this would be translated into:

```
<output ... method="path" location="./nodeid/sampleid/result/TransformParameters.1.txt">
```

## StdoutCollector

The StdoutCollector can collect data from the stdout stream of a program. It filters the stdout line by line matching a predefined regular expression.

The general working is as follows:

1. The location field is taken from the output
2. The substitutions are performed on the location field (see below)
3. The updated location field will be used as a [regular expression](#) filter
4. The stdout is scanned line by line and the [regular expression](#) filter is applied

The special substitutions performed on the location use the Format Specification Mini-Language [Format Specification Mini-Language](#). The predefined fields that can be used are:

- `inputs`, an object with the input values (use like `{inputs.image[0]}`)
- `outputs`, an object with the output values (use like `{outputs.result[0]}`)
- `special` which has two subfields:
  - `special.cardinality`, the index of the current cardinality
  - `special.extension`, is the extension for the output `DataType`

---

**Note:** because the plugin scans line by line, it is impossible to catch multi-line output into a single value

---

## 1.6.2 ExecutionPlugin Reference

This class is the base for all Plugins to execute jobs somewhere. There are many methods already in place for taking care of stuff.

There are fall-backs for certain features, but if a system already implements those it is usually preferred to skip the fall-back and let the external system handle it. There are a few flags to enable/disable these features:

- `cls.SUPPORTS_CANCEL` indicates that the plugin can cancel queued jobs
- `cls.SUPPORTS_HOLD_RELEASE` indicates that the plugin can queue jobs in a hold state and can release them again (if not, the base plugin will create a hidden queue for held jobs). The plugin should respect the `Job.status == JobState.hold` when queueing jobs.
- `cls.SUPPORTS_DEPENDENCY` indicate that the plugin can manage job dependencies, if not the base plugin job dependency system will be used and jobs will only be submitted when all dependencies are met.
- `cls.CANCELS_DEPENDENCIES` indicates that if a job is cancelled it will automatically cancel all jobs depending on that job. If not the plugin traverses the dependency graph and kills each job manually.

---

**Note:** If a plugin supports dependencies it is assumed that when a job gets cancelled, the depending job also gets cancelled automatically!

---

Most plugins should only need to redefine a few abstract methods:

- `__init__` the constructor
- `cleanup` a clean up function that frees resources, closes connections, etc
- `_queue_job` the method that queues the job for execution

Optionally an extra job finished callback could be added:

- `_job_finished` extra callback for when a job finishes

If `SUPPORTS_CANCEL` is set to `True`, the plugin should also implement:

- `_cancel_job` cancels a previously queued job

If `SUPPORTS_HOLD_RELEASE` is set to `True`, the plugin should also implement:

- `_hold_job` hold a job that is currently held
- `_release_job` releases a job that is currently held

If `SUPPORTED_DEPENDENCY` is set to `True`, the plugin should:



- Make sure to use the `Job.hold_jobs` as a list of its dependencies

Not all of the functions need to actually do anything for a plugin. There are examples of plugins that do not really need a `cleanup`, but for safety you need to implement it. Just using a `pass` for the method could be fine in such a case.

**Warning:** When overwriting other functions, extreme care must be taken not to break the plugins working, as there is a lot of bookkeeping that can go wrong.

scheme	ExecutionPlugin
BlockingExecution	<i>BlockingExecution</i>
DRMAAExecution	<i>DRMAAExecution</i>
LinearExecution	<i>LinearExecution</i>
ProcessPoolExecution	<i>ProcessPoolExecution</i>
RQExecution	<i>RQExecution</i>
SlurmExecution	<i>SlurmExecution</i>
StrongrExecution	<i>StrongrExecution</i>

## BlockingExecution

The blocking execution plugin is a special plugin which is meant for debug purposes. It will not queue jobs but immediately execute them inline, effectively blocking `fastr` until the `Job` is finished. It is the simplest execution plugin and can be used as a template for new plugins or for testing purposes.

## DRMAAExecution

A DRMAA execution plugin to execute `Jobs` on a Grid Engine cluster. It uses a configuration option for selecting the queue to submit to. It uses the python `drmaa` package.

---

**Note:** To use this plugin, make sure the `drmaa` package is installed and that the execution is started on an SGE submit host with DRMAA libraries installed.

---



---

**Note:** This plugin is at the moment tailored to SGE, but it should be fairly easy to make different subclasses for different DRMAA supporting systems.

---

## Configuration fields

The following configuration fields are added to the `fastr` config:

name	type	description	default
<code>drmaa_queue</code>	str	The default queue to use for jobs send to the scheduler	'week'
<code>drmaa_max_jobs</code>	int	The maximum jobs that can be send to the scheduler at the same time (0 for no limit)	0
<code>drmaa_engine</code>	str	The engine to use (options: <code>grid_engine</code> , <code>torque</code> )	'grid_engine'
<code>drmaa_job_check_interval</code>	int	The interval in which the job checker will start to check for stale jobs	900
<code>drmaa_num_undetermined_to_fail</code>	int	Number of consecutive times a job state has be undetermined to be considered to have failed	3

## LinearExecution

An execution engine that has a background thread that executes the jobs in order. The queue is a simple FIFO queue and there is one worker thread that operates in the background. This plugin is meant as a fallback when other plugins do not function properly. It does not multi-processing so it is safe to use in environments that do not support that.

## ProcessPoolExecution

A local execution plugin that uses multiprocessing to create a pool of worker processes. This allows fastr to execute jobs in parallel with true concurrency. The number of workers can be specified in the fastr configuration, but the default amount is the number of cores - 1 with a minimum of 1.

**Warning:** The ProcessPoolExecution does not check memory requirements of jobs and running many workers might lead to memory starvation and thus an unresponsive system.

### Configuration fields

The following configuration fields are added to the fastr config:

name	type	description	default
process_pool_worker_number	int	Number of workers to use in a process pool	1

## RQExecution

A execution plugin based on Redis Queue. Fastr will submit jobs to the redis queue and workers will peel the jobs from the queue and process them.

This system requires a running redis database and the database url has to be set in the fastr configuration.

---

**Note:** This execution plugin required the `redis` and `rq` packages to be installed before it can be loaded properly.

---

### Configuration fields

The following configuration fields are added to the fastr config:

name	type	description	default
rq_host	str	The url of the redis serving the redis queue	'redis://localhost:6379/0'
rq_queue	str	The redis queue to use	'default'

## SlurmExecution

The SlurmExecution plugin allows you to send the jobs to SLURM using the sbatch command. It is pure python and uses the sbatch, scancel, squeue and scontrol programs to control the SLURM scheduler.

### Configuration fields

The following configuration fields are added to the fastr config:

name	type	description	de- fault
slurm_job_check_interval	int	The interval in which the job checker will start to check for stale jobs	30
slurm_partition	str	The slurm partition to use	,

## StrongrExecution

NOT DOCUMENTED!

### 1.6.3 FlowPlugin Reference

Plugin that can manage an advanced data flow. The plugins override the execution of node. The execution receives all data of a node in one go, so not split per sample combination, but all data on all inputs in one large payload. The flow plugin can then re-order the data and create resulting samples as it sees fits. This can be used for all kinds of specialized data flows, e.g. cross validation.

To create a new FlowPlugin there is only one method that needs to be implemented: `execute`.

scheme	FlowPlugin
CrossValidation	<i>CrossValidation</i>

## CrossValidation

Advanced flow plugin that generated a cross-validation data flow. The node need an input with data and an input number of folds. Based on that the outputs test and train will be supplied with a number of data sets.

### 1.6.4 IOPlugin Reference

*IOPlugins* are used for data import and export for the sources and sinks. The main use of the *IOPlugins* is during execution (see *Execution*). The *IOPlugins* can be accessed via `fastr.ioplugins`, but generally there should be no need for direct interaction with these objects. The use of is mainly via the URL used to specify source and sink data.

scheme	IOPlugin
CommaSeperatedValueFile	<i>CommaSeperatedValueFile</i>
FileSystem	<i>FileSystem</i>
HTTPPlugin	<i>HTTPPlugin</i>
NetworkScope	<i>NetworkScope</i>
Null	<i>Null</i>
Reference	<i>Reference</i>
S3Filesystem	<i>S3Filesystem</i>
VirtualFileSystem	<i>VirtualFileSystem</i>
VirtualFileSystemRegularExpression	<i>VirtualFileSystemRegularExpression</i>
VirtualFileSystemValueList	<i>VirtualFileSystemValueList</i>
XNATStorage	<i>XNATStorage</i>

## CommaSeperatedValueFile

The CommaSeperatedValueFile an expand-only type of IOPlugin. No URLs can actually be fetched, but it can expand a single URL into a larger amount of URLs.

The `csv://` URL is a `vfs://` URL with a number of query variables available. The URL mount and path should point to a valid CSV file. The query variable then specify what column(s) of the file should be used.

The following variable can be set in the query:

variable	usage
value	the column containing the value of interest, can be int for index or string for key
id	the column containing the sample id (optional)
header	indicates if the first row is considered the header, can be <code>true</code> or <code>false</code> (optional)
delimiter	the delimiter used in the csv file (optional)
quote	the quote character used in the csv file (optional)
reformat	a reformatting string so that <code>value = reformat.format(value)</code> (used before <code>relative_path</code> )
relative_path	indicates the entries are relative paths (for files), can be <code>true</code> or <code>false</code> (optional)

The header is by default `false` if the neither the `value` and `id` are set as a string. If either of these are a string, the header is required to define the column names and it automatically is assumed `true`

The delimiter and quota characters of the file should be detected automatically using the [Sniffer](#), but can be forced by setting them in the URL.

Example of valid csv URLs:

```
# Use the first column in the file (no header row assumed)
csv://mount/some/dir/file.csv?value=0

# Use the images column in the file (first row is assumed header row)
csv://mount/some/dir/file.csv?value=images

# Use the segmentations column in the file (first row is assumed header row)
# and use the id column as the sample id
csv://mount/some/dir/file.csv?value=segmentations&id=id

# Use the first column as the id and the second column as the value
# and skip the first row (considered the header)
csv://mount/some/dir/file.csv?value=1&id=0&header=true

# Use the first column and force the delimiter to be a comma
csv://mount/some/dir/file.csv?value=0&delimiter=,
```

## FileSystem

The FileSystem plugin is create to handle `file://` type or URLs. This is generally not a good practice, as this is not portable over between machines. However, for test purposes it might be useful.

The URL scheme is rather simple: `file://host/path` (see [wikipedia](#) for details)

We do not make use of the `host` part and at the moment only support `localhost` (just leave the host empty) leading to `file:///` URLs.

**Warning:** This plugin ignores the hostname in the URL and does only accept driver letters on Windows in the form `c:/`

## HTTPPlugin

**Warning:** This Plugin is still under development and has not been tested at all. example url: <https://server.io/path/to/resource>

## NetworkScope

A simple source plugin that allows to get data from the Network scope. This uses the `network://` scheme.

An uri of `network://atlases/image_01.nii.gz` would be translated to `vfs://mount/network/atlases/image_01.nii.gz` given that the network would be created/loaded from `vfs://mount/network/networkfile.py`.

**Warning:** This means that the network file must be present in a folder mounted in the `vfs` system. Fastr will use a `vfs` to translate the path between main process and execution workers.

If the resulting uri should be a different `vfs`-based url than the default `vfs://` then a combined scheme can be used. For example `network+vfslist://atlases/list.txt` would be translated into `vfslist://mount/network/atlases/list.txt` and the result would be run by the `vfslist` plugin.

## Null

The Null plugin is create to handle `null://` type or URLs. These URLs are indicating the sink should not do anything. The data is not written to anywhere. Besides the scheme, the rest of the URL is ignored.

## Reference

The Reference plugin is create to handle `ref://` type or URLs. These URLs are to make the sink just write a simple reference file to the data. The reference file contains the `DataType` and the value so the result can be reconstructed. It for files just leaves the data on disk by reference. This plugin is not useful for production, but is used for testing purposes.

## S3Filesystem

**Warning:** As this IOPlugin is under development, it has not been thoroughly tested.  
example url: `s3://bucket.server/path/to/resource`

## VirtualFileSystem

The virtual file system class. This is an IOPlugin, but also heavily used internally in fastr for working with directories. The `VirtualFileSystem` uses the `vfs://` url scheme.

A typical virtual filesystem url is formatted as `vfs://mountpoint/relative/dir/from/mount.ext`

Where the mountpoint is defined in the [Config file](#). A list of the currently known mountpoints can be found in the `fastr.config` object

```
>>> fastr.config.mounts
{'example_data': '/home/username/fastr-feature-documentation/fastr/fastr/examples/data',
 'home': '/home/username/',
 'tmp': '/home/username/FastrTemp'}
```

This shows that a url with the mount home such as `vfs://home/tempdir/testfile.txt` would be translated into `/home/username/tempdir/testfile.txt`.

There are a few default mount points defined by Fastr (that can be changed via the config file).

mountpoint	default location
home	the users home directory ( <code>expanduser('~')</code> )
tmp	the fastr temporary dir, defaults to <code>tempfile.gettempdir()</code>
example_data	the fastr example data directory, defaults <code>\$FASTRDIR/example/data</code>

## VirtualFileSystemRegularExpression

The `VirtualFileSystemValueList` an expand-only type of `IOPlugin`. No URLs can actually be fetched, but it can expand a single URL into a larger amount of URLs.

A `vfsregex://` URL is a `vfs` URL that can contain regular expressions on every level of the path. The regular expressions follow the [re module](#) definitions.

An example of a valid URLs would be:

```
vfsregex://tmp/network_dir/.*/*/__fastr_result__.pickle.gz
vfsregex://tmp/network_dir/nodeX/(?P<id>.*).__fastr_result__.pickle.gz
```

The first URL would result in all the `__fastr_result__.pickle.gz` in the working directory of a Network. The second URL would only result in the file for a specific node (nodeX), but by adding the named group id using `(?P<id>.*)` the sample id of the data is automatically set to that group (see [Regular Expression Syntax](#) under the special characters for more info on named groups in regular expression).

Concretely if we would have a directory `vfs://mount/somedir` containing:

```
image_1/Image.nii
image_2/image.nii
image_3/anotherimage.nii
image_5/inconsistentnamingftw.nii
```

we could match these files using `vfsregex://mount/somedir/(?P<id>image_\d+)/.*\..nii` which would result in the following source data after expanding the URL:

```
{'image_1': 'vfs://mount/somedir/image_1/Image.nii',
 'image_2': 'vfs://mount/somedir/image_2/image.nii',
 'image_3': 'vfs://mount/somedir/image_3/anotherimage.nii',
 'image_5': 'vfs://mount/somedir/image_5/inconsistentnamingftw.nii'}
```

Showing the power of this regular expression filtering. Also it shows how the ID group from the URL can be used to have sensible sample ids.

**Warning:** due to the nature of regexp on multiple levels, this method can be slow when having many matches on the lower level of the path (because the tree of potential matches grows) or when directories that are parts of the path are very large.

### VirtualFileSystemValueList

The VirtualFileSystemValueList is an expand-only type of IOPlugin. No URLs can actually be fetched, but it can expand a single URL into a larger amount of URLs. A `vfslist://` URL basically is a url that points to a file using `vfs`. This file then contains a number of lines each containing another URL.

If the contents of a file `vfs://mount/some/path/contents` would be:

```
vfs://mount/some/path/file1.txt
vfs://mount/some/path/file2.txt
vfs://mount/some/path/file3.txt
vfs://mount/some/path/file4.txt
```

Then using the URL `vfslist://mount/some/path/contents` as source data would result in the four files being pulled.

**Note:** The URLs in a `vfslist` file do not have to use the `vfs` scheme, but can use any scheme known to the Fastr system.

### XNATStorage

**Warning:** As this IOPlugin is under development, it has not been thoroughly tested.

The XNATStorage plugin is an IOPlugin that can download data from and upload data to an XNAT server. It uses its own `xnat://` URL scheme. This is a scheme specific for this plugin and though it looks somewhat like the XNAT rest interface, a different type of URL.

Data resources can be accessed directly by a data url:

```
xnat://xnat.example.com/data/archive/projects/sandbox/subjects/subject001/experiments/
↳ experiment001/scans/T1/resources/DICOM
xnat://xnat.example.com/data/archive/projects/sandbox/subjects/subject001/experiments/*_
↳ BRAIN/scans/T1/resources/DICOM
```

In the second URL you can see a wildcard being used. This is possible as long as it resolves to exactly one item.

The `id` query element will change the field from the default experiment to subject and the `label` query element sets the use of the label as the fastr id (instead of the XNAT id) to `True` (the default is `False`)

To disable `https` transport and use `http` instead the query string can be modified to add `insecure=true`. This will make the plugin send requests over `http`:

```
xnat://xnat.example.com/data/archive/projects/sandbox/subjects/subject001/experiments/*_
↳ BRAIN/scans/T1/resources/DICOM?insecure=true
```

For sinks it is important to know where to save the data. Sometimes you want to save data in a new assessor/resource and it needs to be created. To allow the Fastr sink to create an object in XNAT, you have to supply the type as a query parameter:

```
xnat://xnat.bmia.nl/data/archive/projects/sandbox/subjects/S01/experiments/_BRAIN/
↪assessors/test_assessor/resources/IMAGE/files/image.nii.gz?resource_
↪type=xnat:resourceCatalog&assessor_type=xnat:qcAssessmentData
```

Valid options are: `subject_type`, `experiment_type`, `assessor_type`, `scan_type`, and `resource_type`.

If you want to do a search where multiple resources are returned, it is possible to use a search url:

```
xnat://xnat.example.com/search?projects=sandbox&subjects=subject[0-9][0-9][0-9]&
↪experiments=*_BRAIN&scans=T1&resources=DICOM
```

This will return all DICOMs for the T1 scans for experiments that end with `_BRAIN` that belong to a subjectXXX where XXX is a 3 digit number. By default the ID for the samples will be the experiment XNAT ID (e.g. `XNAT_E00123`). The wildcards that can be used are the same UNIX shell-style wildcards as provided by the module `fnmatch`.

It is possible to change the id to a different fields id or label. Valid fields are `project`, `subject`, `experiment`, `scan`, and `resource`:

```
xnat://xnat.example.com/search?projects=sandbox&subjects=subject[0-9][0-9][0-9]&
↪experiments=*_BRAIN&scans=T1&resources=DICOM&id=subject&label=true
```

The following variables can be set in the search query:

variable	default	usage
projects	*	The project(s) to select, can contain wildcards (see <code>fnmatch</code> )
subjects	*	The subject(s) to select, can contain wildcards (see <code>fnmatch</code> )
experiments	*	The experiment(s) to select, can contain wildcards (see <code>fnmatch</code> )
scans	*	The scan(s) to select, can contain wildcards (see <code>fnmatch</code> )
resources	*	The resource(s) to select, can contain wildcards (see <code>fnmatch</code> )
id	experiment	What field to use as the id, can be: <code>project</code> , <code>subject</code> , <code>experiment</code> , <code>scan</code> , or <code>resource</code>
label	false	Indicate the XNAT label should be used as fastr id, options <code>true</code> or <code>false</code>
insecure	false	Change the url scheme to be used to <code>http</code> instead of <code>https</code>
verify	true	(Dis)able the verification of SSL certificates
regex	false	Change search to use regex <code>re.match()</code> instead of <code>fnmatch</code> for matching
overwrite	false	Tell XNAT to overwrite existing files if a file with the name is already present

For storing credentials the `.netrc` file can be used. This is a common way to store credentials on UNIX systems. It is required that the file is only accessible by the owner only or a `NetrcParseError` will be raised. A `netrc` file is really easy to create, as its entries look like:

```
machine xnat.example.com
  login username
  password secret123
```

See the `netrc` module or the [GNU inet utils website](#) for more information about the `.netrc` file.

**Note:** On windows the location of the `netrc` file is assumed to be `os.path.expanduser('~/_netrc')`. The leading underscore is because windows does not like filename starting with a dot.

**Note:** For scan the label will be the scan type (this is initially the same as the series description, but can be updated manually or the XNAT scan type cleanup).



**Warning:** labels in XNAT are not guaranteed to be unique, so be careful when using them as the sample ID.

For background on XNAT, see the [XNAT API DIRECTORY](#) for the REST API of XNAT.

## 1.6.5 Interface Reference

Abstract base class of all Interfaces. Defines the minimal requirements for all Interface implementations.

scheme	Interface
FastrInterface	<i>FastrInterface</i>
FlowInterface	<i>FlowInterface</i>
NipypeInterface	<i>NipypeInterface</i>

### FastrInterface

The default Interface for fastr. For the command-line Tools as used by fastr. It build a commandline call based on the input/output specification.

The fields that can be set in the interface:

Attribute		Description
id		The id of this Tool (used internally in fastr)
inputs[]		List of Inputs that can are accepted by the Tool
	id	ID of the Input
	name	Longer name of the Input (more human readable)
	datatype	The ID of the DataType of the Input <sup>1</sup>
	enum[]	List of possible values for an EnumType (created on the fly by fastr) <sup>Page 54, 1</sup>
	prefix	Commandline prefix of the Input (e.g. -in, -i)
	cardinality	Cardinality of the Input
	repeat_prefix	Flag indicating if for every value of the Input the prefix is repeated
	required	Flag indicating if the input is required
	nospace	Flag indicating if there is no space between prefix and value (e.g. -in=val)
	format	For DataTypes that have multiple representations, indicate which one to use
	default	Default value for the Input
	description	Long description for an input
outputs[]		List of Outputs that are generated by the Tool (and accessible to fastr)
	id	ID of the Output

continues on next page

Table 1.1 – continued from previous page

Attribute		Description
	<code>name</code>	Longer name of the Output (more human readable)
	<code>datatype</code>	The ID of the DataType of the Output <sup>Page 54, 1</sup>
	<code>enum[]</code>	List of possible values for an Enum-Type (created on the fly by fastr) <sup>1</sup>
	<code>prefix</code>	Commandline prefix of the Output (e.g. <code>-out</code> , <code>-o</code> )
	<code>cardinality</code>	Cardinality of the Output
	<code>repeat_prefix</code>	Flag indicating if for every value of the Output the prefix is repeated
	<code>required</code>	Flag indicating if the input is required
	<code>nospace</code>	Flag indicating if there is no space between prefix and value (e.g. <code>-out=val</code> )
	<code>format</code>	For DataTypes that have multiple representations, indicate which one to use
	<code>description</code>	Long description for an input
	<code>action</code>	Special action (defined per DataType) that needs to be performed before creating output value (e.g. <code>'ensure'</code> will make sure an output directory exists)
	<code>automatic</code>	Indicate that output doesn't require commandline argument, but is created automatically by a Tool <sup>2</sup>
	<code>method</code>	The collector plugin to use for the gathering automatic output, see the <a href="#">Collector plugins</a>
	<code>location</code>	Definition where to an automatically, usage depends on the <code>method</code> <sup>2</sup>

## FlowInterface

The Interface use for AdvancedFlowNodes to create the advanced data flows that are not implemented in the fastr. This allows nodes to implement new data flows using the plugin system.

The definition of FlowInterfaces are very similar to the default FastrInterfaces.

---

**Note:** A flow interface should be using a specific FlowPlugin

---

<sup>1</sup> `datatype` and `enum` are conflicting entries, if both specified `datatype` has presedence

<sup>2</sup> More details on defining automatica output are given in [TODO]

## NipypeInterface

Experimental interfaces to using nipype interfaces directly in fastr tools, only using a simple reference.

To create a tool using a nipype interface just create an interface with the correct type and set the nipype argument to the correct class. For example in an xml tool this would become:

```
<interface class="NipypeInterface">
  <nipype_class>nipype.interfaces.elastix.Registration</nipype_class>
</interface>
```

**Note:** To use these interfaces nipype should be installed on the system.

**Warning:** This interface plugin is basically functional, but highly experimental!

## 1.6.6 ReportingPlugin Reference

Base class for all reporting plugins. The plugin has a number of methods that can be implemented that will be called on certain events. On these events the plugin can inspect the presented data and take reporting actions.

scheme	ReportingPlugin
ElasticsearchReporter	<i>ElasticsearchReporter</i>
PimReporter	<i>PimReporter</i>
SimpleReport	<i>SimpleReport</i>

### ElasticsearchReporter

NOT DOCUMENTED!

#### Configuration fields

The following configuration fields are added to the fastr config:

name	type	description	default
elasticsearch_host	str	The elasticsearch host to report to	''
elasticsearch_index	str	The elasticsearch index to store data in	'fastr'
elasticsearch_debug	bool	Setup elasticsearch debug mode to send stdout stderr on job succes	False

### PimReporter

NOT DOCUMENTED!

#### Configuration fields

The following configuration fields are added to the fastr config:

name	type	description	default
pim_host	str	The PIM host to report to	''
pim_username	str	Username to send to PIM	Username of the currently logged in user
pim_update_interval	float	The interval in which to send jobs to PIM	2.5
pim_batch_size	int	Maximum number of jobs that can be send to PIM in a single interval	100
pim_debug	bool	Setup PIM debug mode to send stdout stderr on job success	False
pim_finished_timeout	int	Maximum number of seconds after the network finished in which PIM tries to synchronize all remaining jobs	10

## SimpleReport

NOT DOCUMENTED!

## 1.6.7 Target Reference

The abstract base class for all targets. Execution with a target should follow the following pattern:

```
>>> with Target() as target:
...     target.run_command(['sleep', '10'])
```

The Target context operator will set the correct paths/initialization. Within the context command can be ran and when leaving the context the target reverts the state before.

scheme	Target
DockerTarget	<i>DockerTarget</i>
LocalBinaryTarget	<i>LocalBinaryTarget</i>
MacroTarget	<i>MacroTarget</i>
SingularityTarget	<i>SingularityTarget</i>

## DockerTarget

A tool target that is located in a Docker images. Can be run using docker-py. A docker target only need two variables: the binary to call within the docker container, and the docker container to use.

```
{
  "arch": "*",
  "os": "*",
  "binary": "bin/test.py",
  "docker_image": "fastr/test"
}
```

```
<target os="*" arch="*" binary="bin/test.py" docker_image="fastr/test">
```

## LocalBinaryTarget

A tool target that is a local binary on the system. Can be found using

environmentmodules or a path on the executing machine. A local binary target has a number of fields that can be supplied:

- **binary** (required): the name of the binary/script to call, can also be called `bin` for backwards compatibility.
- **modules**: list of modules to load, this can be environmentmodules or lmod modules. If modules are given, the paths, environment\_variables and initscripts are ignored.
- **paths**: a list of paths to add following the structure `{"value": "/path/to/dir", "type": "bin"}`. The types can be `bin` if it should be added to `$PATH` or `lib` if it should be added to the library path (e.g. `$LD_LIBRARY_PATH` for linux).
- **environment\_variables**: a dictionary of environment variables to set.
- **initscript**: a list of script to run before running the main tool
- **interpreter**: the interpreter to use to call the binary e.g. `python`

The LocalBinaryTarget will first check if there are modules given and the module subsystem is loaded. If that is the case it will simply unload all current modules and load the given modules. If not it will try to set up the environment itself by using the following steps:

1. Prepend the bin paths to `$PATH`
2. Prepend the lib paths to the correct environment variable
3. Setting the other environment variables given (`$PATH` and the system library path are ignored and cannot be set that way)
4. Call the initscripts one by one

The definition of the target in JSON is very straightforward:

```
{
  "binary": "bin/test.py",
  "interpreter": "python",
  "paths": [
    {
      "type": "bin",
      "value": "vfs://apps/test/bin"
    },
    {
      "type": "lib",
      "value": "./lib"
    }
  ],
  "environment_variables": {
    "othervar": 42,
    "short_var": 1,
    "testvar": "value1"
  },
  "initscripts": [
    "bin/init.sh"
  ],
  "modules": ["elastix/4.8"]
}
```

In XML the definition would be in the form of:

```
<target os="linux" arch="*" modules="elastix/4.8" bin="bin/test.py" interpreter="python">
  <paths>
    <path type="bin" value="vfs://apps/test/bin" />
    <path type="lib" value="./lib" />
  </paths>
  <environment_variables short_var="1">
    <testvar>value1</testvar>
    <othervar>42</othervar>
  </environment_variables>
  <initscripts>
    <initscript>bin/init.sh</initscript>
  </initscripts>
</target>
```

### MacroTarget

A target for MacroNodes. This target cannot be executed as the MacroNode handles execution differently. But this contains the information for the MacroNode to find the internal Network.

### SingularityTarget

A tool target that is run using a singularity container, see the [singularity website](#)

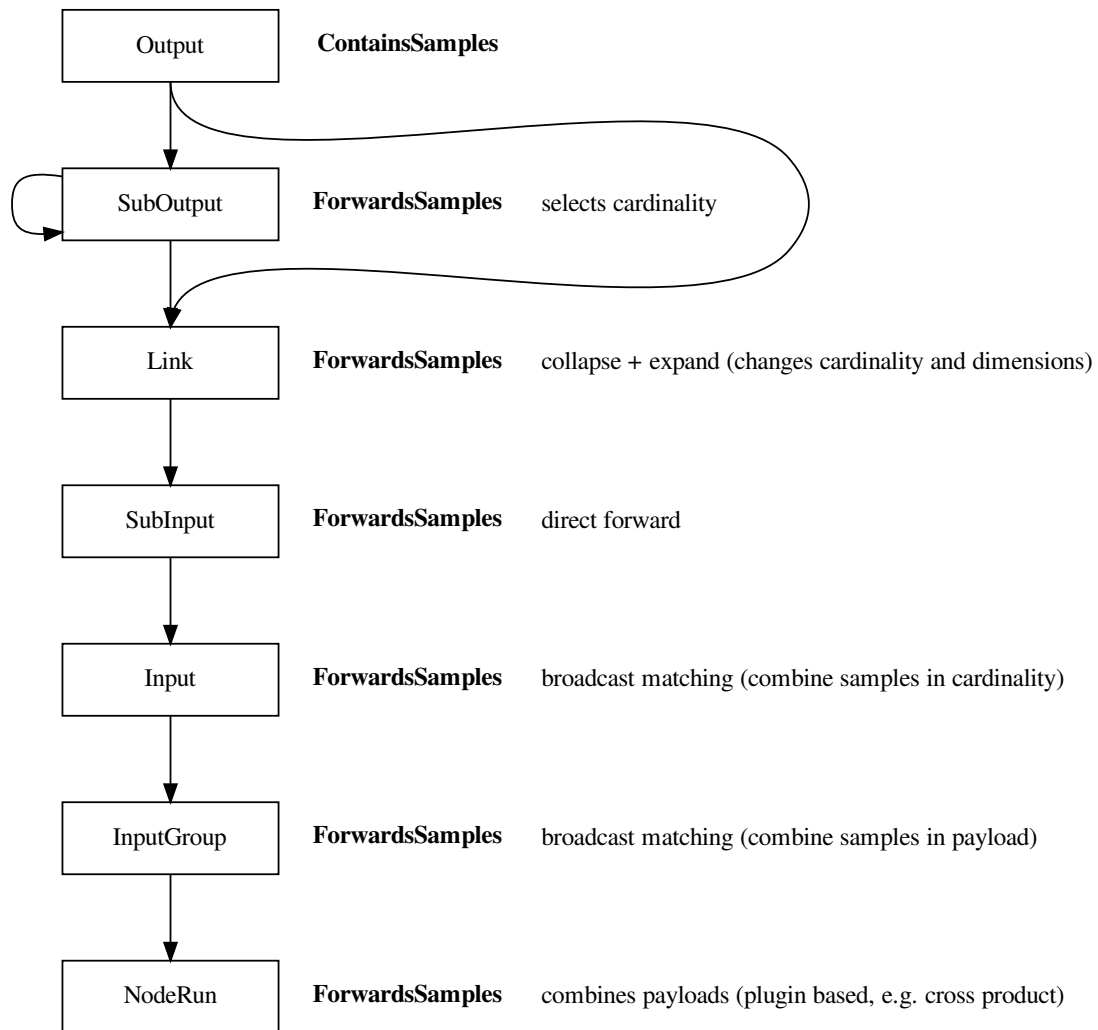
- **binary** (required): the name of the binary/script to call, can also be called **bin** for backwards compatibility.
- **container** (required): the singularity container to run, this can be in url form for singularity pull or as a path to a local container
- **interpreter**: the interpreter to use to call the binary e.g. python

## 1.7 Development and Design Documentation

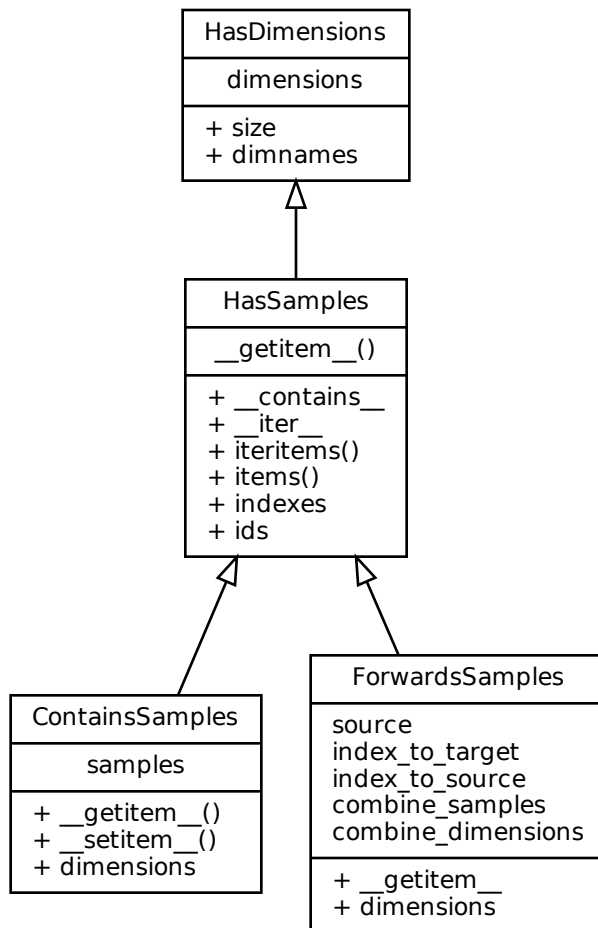
In this chapter we will discuss the design of Fastr in more detail. We give pointers for development and add the design documents as we currently envision Fastr. This is both for people who are interested in the Fastr develop and for current developers to have an archive of the design decision agreed upon.

### 1.7.1 Sample flow in Fastr

The current Sample flow is the following:



The idea is that we make a common interface for all classes that are related to the flow of Samples. For this we propose the following mixin classes that provide the interface and allow for better code sharing. The basic structure of the classes is given in the following diagram:



The abstract and mixin methods are as follows:



ABC	Inherits from	Abstract Methods	Mixin methods
HasDimensions		dimensions	size dimnames
HasSamples	HasDimensions	__getitem__	__contains__ __iter__ iteritems items indexes ids
ContainsSamples	HasSamples	samples	__getitem__ __setitem__ dimensions
ForwardsSamples	HasSamples	source index_to_target index_to_source combine_samples combine_dimensions	__getitem__ dimensions

---

**Note:** Though the flow is currently working like this, the mixins are not yet created.

---

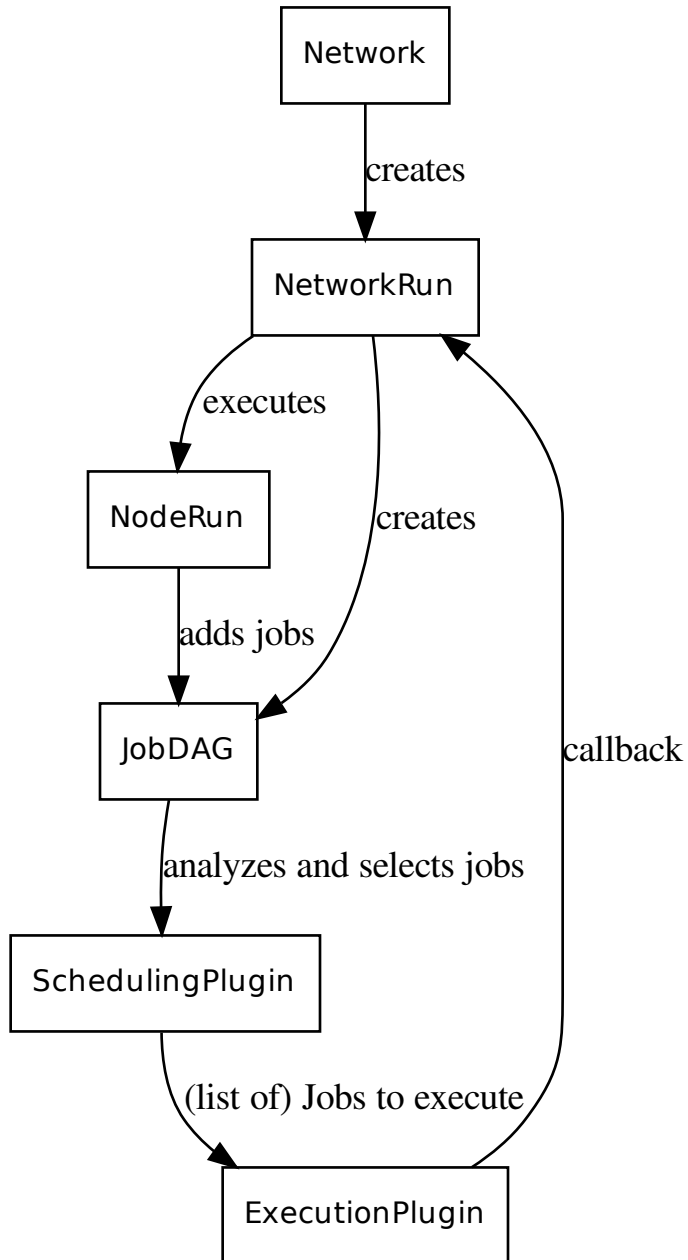
## 1.7.2 Network Execution

The network execution should contain a number of steps:

- **Network**
  - Creates a `NetworkRun` based on the current layout
- **NetworkRun**
  - Transform the `Network` (possibly joining `Nodes` of certain interface into a combined `NodeRun` etc)
  - Start generation of the Job Direct Acyclic Graph (DAG)
- **SchedulingPlugin**
  - Prioritize `Jobs` based on some predefined rules
  - Combine certain `Jobs` to improve efficiency (e.g. minimize i/o on a grid)
- **ExecutionPlugin**
  - Run a (list of) `Jobs`. If there is more than one jobs, run them sequentially on same execution host using a local temp for intermediate files.

- On finished callback: Updated DAG with newly ready jobs, or remove cancelled jobs

This could be visualized as the following loop:



The callback of the ExecutionPlugin to the NetworkRun would trigger the execution of the relevant NodeRuns and the addition of more Jobs to the JobDAG.

---

**Note:** The Job DAG should be thread-safe as it could be both read and extended at the same time.

---

---

**Note:** If a list of jobs is send to the `ExecutionPlugin` to be run as on Job on an external execution platform, the resources should be combined as follows: memory=max, cores=max, runtime=sum

---

---

**Note:** If there are execution hosts that have mutliple cores the `ExecutionPlugin` should manage this (for example by using pilot jobs). The `SchedulingPlugin` creates units that should be run sequentially on the resources noted and will not attempt parallelization

---

A `NetworkRun` would be contain similar information as the `Network` but not have functionality for editing/changing it. It would contain the functionality to execute the `Network` and track the status and samples. This would allow `Network.execute` to create multiple concurrent runs that operate indepent of each other. Also editing a `Network` after the run started would have no effect on that run.

---

**Note:** This is a plan, not yet implemented

---

---

**Note:** For this to work, it would be important for a Jobs to have forward and backward dependency links.

---

## SchedulingPlugins

The idea of the plugin is that it would give a priority on Jobs created by a `Network`. This could be done based on different strategies:

- Based on (sorted) sample id's, so that one sample is always prioritized over others. The idea is that samples are process as much as possible in order, finishing the first sample first. Only processing other samples if there is left-over capacity.
- Based on distance to a (particular) Sink. This is to generate specific results as quick as possible. It would not focus on specific samples, but give priority to whatever sample is closest to being finished.
- Based on the distance to from a Souce. Based on the sign of the weight it would either keep all samples on the same stage as much as possible, only progressing to a new `NodeRun` when all samples are done with the previous `NodeRun`, or it would push samples with accelerated rates.

Additionally it will group Jobs to be executed on a single host. This could reduce i/o and limited the number of jobs an external scheduler has to track.

---

**Note:** The interface for such a plugin has not yet been established.

---

## 1.7.3 Secrets

“Something that is kept or meant to be kept unknown or unseen by others.”

### Using secrets

Fastr IOPlugins that need authentication data should use the Fastr SecretService for retrieving such data. The Secret-Service can be used as follows.

```
from fastr.utils.secrets import SecretService
from fastr.utils.secrets.exceptions import CouldNotRetrieveCredentials

secret_service = SecretService()

try:
    password = secret_service.find_password_for_user('testserver.lan:9000', 'john-doe')
except CouldNotRetrieveCredentials:
    # the password was not found
    pass
```

### Implementing a SecretProvider

A SecretProvider is implemented as follows:

1. Create a file in fastr/utils/secrets/providers/<yourprovidername>.py
2. Use the template below to write your SecretProvider
3. Add the secret provider to fastr/utils/secrets/providers/\_\_init\_\_.py
4. Add the secret provider to fastr/utils/secrets/secretservice.py: import it and add it to the array in function `_init_providers`

```
from fastr.utils.secrets.secretprovider import SecretProvider
from fastr.utils.secrets.exceptions import CouldNotRetrieveCredentials,
↳ CouldNotSetCredentials, CouldNotDeleteCredentials, NotImplemented

try:
    # this is where libraries can be imported
    # we don't want fastr to crash if a specific
    # library is unavailable
    # import my-library
except (ImportError, ValueError) as e:
    pass

class KeyringProvider(SecretProvider):
    def __init__(self):
        # if libraries are imported in the code above
        # we need to check if import was successful
        # if it was not, raise a RuntimeError
        # so that FASTR ignores this SecretProvider
        # if 'my-library' not in globals():
        #     raise RuntimeError("my-library module required")
```

(continues on next page)

(continued from previous page)

```

pass

def get_password_for_user(self, machine, username):
    # This function should return the password as a string
    # or raise a CouldNotRetrieveCredentials error if the password
    # is not found.
    # In the event that this function is unsupported a
    # NotImplemented exception should be thrown
    raise NotImplemented()

def set_password_for_user(self, machine, username, password):
    # This function should set the password for a specified
    # machine + user. If anything goes wrong while setting
    # the password a CouldNotSetCredentials error should be raised.
    # In the event that this function is unsupported a
    # NotImplemented exception should be thrown
    raise NotImplemented()

def del_password_for_user(self, machine, username):
    # This function should delete the password for a specified
    # machine + user. If anything goes wrong while setting
    # the password a CouldNotDeleteCredentials error should be raised.
    # In the event that this function is unsupported a
    # NotImplemented exception should be thrown
    raise NotImplemented()

```

## 1.8 Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](#) and this project adheres to [Semantic Versioning](#)

### 1.8.1 3.3.1 - 2022-10-13

#### Added

- NetworkRun keeps track of the directory (scope) from which they were loaded
- A `network://` ioplugin that allows user to retrieve data relative to a network file
- Added the `examples` and `fastr_home` default mounts to the configuration
- Added support for GPUs and a memory multiplier in the ResourceLimit class. Limits and requirements set for the use of GPUs in ResourceLimit are now supported in the Slurm execution plugin.

## Fixed

- Deferred without valid target could cause errors in checksum and validate
- Python 3.10 compatability fix
- Jobs are now submitted during creation and not after all jobs have been generated

## 1.8.2 3.3.0 - 2021-06-11

### Added

- Added concept of missing data. Using *fastr.MISSING* as source data will note a sample as missing. The network will run each job that has missing data will not be executed and their outputs will be noted as missing too. At the sink the result will be set to missing instead of failed, allowing for a partial execution of a network if it is known upfront some data is missing.
- Added functionality for creating a reference result for Tool verification, in the form of `fastr.utils.verify.create_tool_test()`.
- *tracking\_id* argument to a network run which gives the run a tracking id, all log messages from the network run will be tagged with the tracking id for filtering/combining the logs in a central log system

### Fixes

- Fixes *fastr verify* for Tools by minor changes in `Tool.test_tool()`.
- Fixes bug in Slurm execution plugin
- Fixes in `ProcessPoolExecutor` with the cleanup etc
- Fixes bug with setting environment variable in `LocalBinaryTarget`
- Fixes issue with case-sensitivity in vfs on Windows

## 1.8.3 3.2.3 - 2020-06-25

### Fixed

- Warning for non-production environment didn't handle git tag correctly

## 1.8.4 3.2.2 - 2020-06-25

### Fixed

- Fixed a bug where `ConstantNodes` would not always set their data to use a `DataType` subclass.
- Made version system scrape info from git instead of mercurial to reflect the change in versioning system.

### 1.8.5 3.2.1 - 2020-06-22

#### Fixed

- Some bugs on windows due to use of Path in subprocess arguments
- Added retry to serializable in case of small filesystem sync/timing errors

### 1.8.6 3.2.0 - 2020-06-19

#### Changed

- Changed serialization in Fastr. Networks and Jobs have a better format and are serialized to yaml by default. This makes the job files human readable.

### 1.8.7 3.1.4 - 2020-06-10

#### Added

- Added functionality to be able to use the cardinality of one of the items in an ordereddict input or output.
- Added dependency list function to the Network API.

### 1.8.8 3.1.3 - 2019-11-28

#### Added

- Support for FASTR\_CONFIG\_DIRS to add extra configuration directories (they will be loaded in order after the config.d directory has been loaded).

#### Improved

- The DRMAA execution plugin is more robust and less likely to encounter errors that will cause the execution to become stuck.

#### Fixed

- Bugs in `file://` IOPlugin

### 1.8.9 3.1.2 - 2019-06-18

#### Improved

- Avoid execution plugins calling cleanup multiple times
- Tools can now set an input to environment variables using the `environ` attribute. The parameter will NOT be put command-line anymore and instead be dispatched via an environment variable given by the `environ` argument value

## **Fixed**

- Bug in XNATStorage plugin where files with a path within the resource could not be correctly located
- Add timeout when waiting to send to PIM
- Fix problem with non-requested outputs being able to invalidate a job execution

## **1.8.10 3.1.1 - 2019-05-02**

## **Fixed**

- Packaging problem in release (old file left in build folder)

## **1.8.11 3.1.0 - 2019-05-02**

## **Added**

- Added support for tools in YAML
- `fastr upgrade` can also upgrade tools from XML to YAML
- `fastr report` command to print an overview report of a job result

## **Fixed**

- Re-added support for named sub-inputs

## **Improved**

- Fixes in `fastr upgrade` to handle more exotic whitespace and arguments
- Small documentation fixes (especially in configuration section)
- Better windows support (tested by users)

## **Changed**

- In ResourceLimits the default time of jobs is now None (no limit) instead of 1 hour.
- By default do not log to files (we noticed fastr logs are not very often read by users and they could cause some issues with log rotation, by default logging to files is turned off, switching it back on can be done by setting `log_to_file = True` in the `fastr.config`)



### 1.8.12 3.0.1 - 2019-03-28

#### Fixed

- Improved implementation of `fastr upgrade` to handle newlines in the `create_node` function properly. Also can handle old-fashioned use of `fastr.toollist[...]` in `create_node`.

### 1.8.13 3.0.0 - 2019-03-05

#### Changed

- Now ported to Python 3.6+ (Python 2 is no longer supported!)
- New public API which is not fully compatible with `fastr 2.x`, the changes are small. The new API will be guaranteed in next minor version upgrades and is considered to be stable.
- Clear way of defining resource limits for Nodes in a Network using the `ResourceLimit` class.
- The datatype and cardinality of inputs of a tool are now checked before the tool is to be executed as an extra safety.
- Dimensions are drawn by default in `network.draw`
- The api now accepts types other than `Output`, `list`, `tuple` when creating a link. When a single value is given it is assumedly a constant from the network definition.
- Drawing a network will not create temporary `.dot` files anymore
- Sinkdata can be a string, in that case it will be the same string for all sink nodes so a `{node}` substitution should be used in the template
- Make the `xnat ioplug` use `xnat+http://` and `xnat+https://` url schemes in favour of `xnat://` with `?insecure=...` (old behaviour will also work for now)
- Complete rewrite of PIM plugin (`PIMReporter`) making use of the new Reporter plugin infrastructure. It also caches all communication with PIM to be resilient against connection interruptions.

#### Added

- `fastr upgrade` command to automatically upgrade a network creation file from `fastr 2.x` to `fastr 3.x` API.
- `http(s)` IOPlugin for downloading files via `http(s)`
- `network.draw` now has a flag to hide the unconnected inputs and output of a node. The unconnected inputs/outputs are hidden by default.
- Reporting plugins, `Fastr` now exposes a number of message hooks which can be listened to by Reporter plugins.

### **Fixed**

- Fixed some bugs with drmaa communication (more safeties added)
- Fixed a bug in the MacroNode update function which could cause networks with MacroNodes to be invalid
- The margins and font size of the network.draw graph rendering are set a bit wider and smaller (resp.) to avoid excessive text overflow.
- Fixed bug in provenance which did not properly chain the provenance of subsequent jobs.

## **1.8.14 2.1.2 - 2018-10-24**

### **Added**

- Allow overriding the timestamp of the network execution

### **Changed**

- Updated PIM publisher to support the new PIM API v2
- Updated XNAT IOPlugin to not crash when creating a resource failed because another process already did that (race condition)
- Make default resource limits for DRMAA configurable
- Add stack trace to FastrExceptions

## **1.8.15 2.1.1 - 2018-06-29**

### **Fixed**

- Fixed some issues with the type estimation of outputs of Jobs and update validation functions of NIFTI files

## **1.8.16 2.1.0 - 2018-04-13**

### **Added**

- SLURM execution plugin based on sbatch, scancel, scontrol and squeue. The plugin supports job dependencies and cancellation.
- Support for running tools in Docker containers using a DockerTarget
- Support for running tools in Singularity containers using a SingularityTarget
- Support for datatypes with multiple extensions (e.g. .tif and .tiff) by setting the extension to a tuple of options. The first extension is leading for deciding filenames in a sink.

## Changed

- Source jobs now also validate the output (and do not only rely on the stderr of the tool)
- Added `preferred_types` attribute to `TypeGroups` that gives the order of preference of members, alternatively the order of `_members` is used (this should be given as tuple or list to be meaningful)
- In the `config.py` you can now access the `USER_DIR` and `SYSTEM_DIR` variables for use in setting other variables. These are only read and changing them will only change subsequent config reads but not the main config values.
- checksum for `nii.gz` now takes the md5 checksum of the decompressed data
- Serialization of `MacroNodes` now should function properly

## Fixed

- BUG in XNAT plugin that made it impossible to download data from scans without an empty type string
- BUG where the order of `OrderedDict` in a source was not preserved
- BUG where newer Werkzeug version requires the web port to be an integer

### 1.8.17 2.0.1 - 2017-10-19

- Fix a bug in the validation of `FilePrefix` datatypes

### 1.8.18 2.0.0 - 2017-09-28

## Added

- The default python logger can now be configured from the `fastr` config file under key `logging_config`
- Support for `MacroNodes`, a `Network` can be used as a `Node` inside of another `Network`. There is should be no limitation on the internal `Network` used, but currently the `MacroNode` ignores `input_groups` on its inputs.
- A sync helper was added to assist in slow file synchronisation over NFS
- Source and Sink can now handle S3 URL's
- `FastrInterface` can now forward errors from a subprocess if they are dumped to stdout or stderr in a json identified by `__FASTR_ERRORS__ = []`.
- A `specials.workdir` field in the location field of automatic outputs that gives the current working directory (e.g. job directory)
- Added support for Torque (using `pbs-drmaa` library) to `DRMAAExecution`
- Added option to set a limit for number of jobs submitted at same time be the `DRMAAExecution`
- Use of the `~/fastr/config.d` directory for adding additional config files. Any `.py` file in there will be parsed in alphabetical order.
- `XNATStorageIOPlugin` now has a retry scheme for uploads, if an uploaded file could not be found on the server, it is retried up to 3 times.
- Added `fastr dump` command to create a zip containing all important debugging information.

## Changed

- FilePrefix type does not have an extension anymore (avoids ugly dot in middle of filename)
- Allow expanding of link where samples have a non-uniform cardinality. This will not result in a sparse array.
- The default for `required` for the automatic outputs is now `False`
- Removed `testtool` commandline subcommand in favour of the `test` subcommand which can test both Tools and Networks
- Moved `nodegroup` specification into the `Node` for speedup

## Fixed

- Stop Jobs from failing when a non-required, non-requested output is invalid
- Bug in boolean value parsing in the Boolean datatype
- Bug in target that caused paths not to be expanded properly in some cases
- Made sure failed sources also create a sample so the failure becomes visible and traceable.
- Bug in XNAT IOPlugin that made download from XNAT seem to fail (while getting the correct data).

## Removed

- `fastr.current_network` has been removed as it was deemed to “magical” and could change things out of the sight of the user.

## 1.8.19 1.2.2 - 2017-08-24

### Fixed

- Fixed a bug breaking the XNAT IOPlugin due to an `xnatpy` version update.

## 1.8.20 1.2.1 - 2017-04-04

### Added

- A `FastrInterface` can now specify a `negate` flag on an automatic output that also has a prefix, which will negate the flag. This is useful for flag the suppress the creation of an output (e.g. `no_mask`). An example is given in the Tool `fastr.util.AutoPrefixNegateTest`.

## Changed

- The provenance and extra information of a Job now is not serialized in the Job, but exported to separate files next to the job file `__fastr_prov__.json` and `__fastr_extra_job_info__.json` which makes the information more accessible and reduces the memory footprint of the main process hugely as it will not read this information back anymore.
- Most execution plugin will not overwrite the `executionscript` stdout and stderr but rather append it. This is only relevant when continuing a run in the an existing temporary directory, but avoids loss of information.

## Fixed

- Bug that stopped the `Link.append` function from returning the newly created link
- Bugs that caused some cardinality computations of the output to fail during execution
- Bug in the `job.tmpurl` that caused double slashes somewhere. Some tools chocked on this when it was used for parameters.

## 1.8.21 1.2.0 - 2017-03-15

### Added

- Failed sample annotation: when a job fails, the result is annotated and forwarded until a `SinkNode`, where we can determine the status and possibly point of failure of the `Sample`.
- Commandline tool `fastr trace` that can inspect a workflow run and help trace errors and print debug information
- Supported for `Lmod` modules environment next to the old `environmentmodules`
- `BaseDataType` descendants are now (un)picklable (including `EnumTypes`)
- Option to use `{extension}` field in `sink_data`, which differs from `{ext}` in that it doesn't include a leading dot.
- Support for Docker targets. A Docker target will execute a command inside of a specified docker container, allowing Tools to use Docker for distribution
- Using the right and left shift operator (`<<` and `>>`) for creating links to Inputs using `input << output` or `output >> input`.
- In the `FastrInterfaces`, automatic outputs can have a prefix for a flag that should be set for the output to be actually generated.
- `Fastr` is now able to limit the amount of `SourceJobs` that are allowed to run concurrently.
- Ability to report progress to PIM (use the `pim_host` field in the config)

### Changed

- Version can now also accept a format based on a date (e.g. `2017-02-17_bananas`) which will be parsed the same way as `2017.02.17_bananas`
- Work on the `ExecutionPlugin` and the corresponding API. Has better fall-backs and a mechanism to advertise plugin capabilities.
- The collector plugins have the `input` and `input_parts` fields merged, and the `output` and `output_parts` fields merged.

## **Fixed**

- In some cases the log directory was not created properly, causing an handled exception
- A bug making the handling of Booleans incorrect for the `FastrInterface`, when a Boolean was given a flag would also appear when it was `False`
- Serialization of the namespace of a `Network` was not correct
- Check version of `Fastr` that creates and executes a `Job` against each other
- `load_gpickle` helper can handle data with `Enums` that use to cause an `AttributeError`
- Output validation of `Jobs` did not work correctly for automatic outputs

### **1.8.22 1.1.2 - 2016-12-22**

## **Fixed**

- The example network in `resources/networks/add_ints.json` was using an old serialization format making it non-functions. Replaced by a new network file.

### **1.8.23 1.1.1 - 2016-12-22**

## **Fixed**

- Network runs called from an interpreter (and not file) caused a crash because the network tried to report the file used. Better handling of these situations.

### **1.8.24 1.1.0 - 2016-12-08**

## **Added**

- Namespaces for resources (tools and networks)
- Network manager located at `fastr.networklist`
- `RQExecution` plugin. This plugin uses `python-rq` to manage a job queue.
- `LinearExecution` plugin. This plugin uses a background thread for execution.
- `BlockingExecution` plugin. This plugin executes jobs in a blocking fashion.
- Automatic generation of documentation for all plugins, the configuration fields and all commandline tools.

## **Changed**

- Provenance is updated with a network dump and used tool definitions.
- New configuration system that uses python files
- New plugin system that integrates with the new configuration system and enables automatic importing of plugins
- The `fastr` command line tools now use an entrypoint which is located in `fastr.utils.cmd`. This code also dispatches the sub commands.

### Removed

- fastr.config file. This is replaced by the config.py file. Go to the docs!

### Fixed

- Adds explicit tool namespace and version to the provenance document.





## FASTR USER REFERENCE

### 2.1 Fastr User Reference

#### fastr.tools

A ToolManager containing all versions of all Tools loaded into the FASTR environment. The ToolManager can be indexed using the Tool id string or a tool id string and a version. For example if you have two versions (4.5 and 4.8) of a tool called *Elastix*:

```
>>> fastr.tools['elastix.Elastix']
Tool Elastix v4.8 (Elastix Registration)
```

Inputs		Outputs
-----		
↪-----		
fixed_image	(ITKImageFile)	directory_
↪(Directory)		
moving_image	(ITKImageFile)	transform_
↪(ElastixTransformFile)		
parameters	(ElastixParameterFile)	log_file _
↪(ElastixLogFile)		
fixed_mask	(ITKImageFile)	
moving_mask	(ITKImageFile)	
initial_transform	(ElastixTransformFile)	
priority	(__Elastix_4.8_interface__priority__Enum__)	
threads	(Int)	

```
>>> fastr.tools['elastix.Elastix', '4.5']
Tool Elastix v4.5 (Elastix Registration)
```

Inputs		Outputs
-----		
↪-----		
fixed_image	(ITKImageFile)	directory_
↪(Directory)		
moving_image	(ITKImageFile)	transform_
↪(ElastixTransformFile)		
parameters	(ElastixParameterFile)	log_file _
↪(ElastixLogFile)		
fixed_mask	(ITKImageFile)	
moving_mask	(ITKImageFile)	
initial_transform	(ElastixTransformFile)	
priority	(__Elastix_4.5_interface__priority__Enum__)	

(continues on next page)

(continued from previous page)

threads	(Int)	
---------	-------	--

### **fastr.types**

A dictionary containing all types loaded into the FASTR environment. The keys are the typenames and the values are the classes.

### **fastr.networks**

A dictionary containing all networks loaded in fastr

**api.create\_network**(*version=None*)

Create a new Network object

#### **Parameters**

- **id** (*str*) – id of the network
- **version** (*Union[Version, str, None]*) – version of the network

#### **Return type**

*Network*

#### **Returns**

**api.create\_network\_copy**()

Create a network based on another Network state. The network state can be a Network or the state gotten from a Network with `__getstate__`.

#### **Parameters**

**network\_state** (*Union[Network, Network, dict]*) – Network (state) to create a copy of

#### **Return type**

*Network*

#### **Returns**

The rebuilt network

**class** **fastr.api.Network**(*id, version=None*)

Representation of a Network for the creating and adapting Networks

**create\_constant**(*datatype, data, id=None, step\_id=None, resources=None, node\_group=None*)

Create a ConstantNode in this Network. The Node will be automatically added to the Network.

#### **Parameters**

- **datatype** (*Union[BaseDataType, str]*) – The DataType of the constant node
- **data** (*Dict[str, Union[List[Union[Tuple[Union[str, int, float], ...], str, int, float]], Mapping[str, Union[Tuple[Union[str, int, float], ...], str, int, float]], Tuple[Union[str, int, float], ...], str, int, float]]]*) – The data to hold in the constant node
- **id** (*Optional[str]*) – The id of the constant node to be created
- **step\_id** (*Optional[str]*) – The step to add the created constant node to
- **resources** (*Optional[ResourceLimit]*) – The resources required to run this node
- **node\_group** (*Optional[str]*) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.

#### **Return type**

*Node*

**Returns**

the newly created constant node

**create\_link**(*source*, *target*, *id=None*, *collapse=None*, *expand=False*)

Create a link between two Nodes and add it to the current Network.

**Parameters**

- **source** (`Union[Input, BaseInput]`) – the output that is the source of the link
- **target** (`Union[Output, BaseOutput]`) – the input that is the target of the link
- **id** (`Optional[str]`) – the id of the link
- **collapse** (`Optional[Tuple[Union[int, str], ...]]`) – The dimensions to collapse in this link.
- **expand** (`bool`) – Flag to expand cardinality into a new dimension

**Return type**

`Link`

**Returns**

the created link

**create\_macro**(*network*, *id=None*)

Create macro node (a node which actually contains a network used as node inside another network).

**Parameters**

- **network** (`Union[Network, Network, dict, Tool, str]`) – The network to use, this can be a network (state), a macro tool, or the path to a python file that contains a function `create_network` which returns the desired network.
- **id** (`Optional[str]`) – The id of the node to be created

**Return type**

`Node`

**Returns**

the newly created node

**create\_node**(*tool*, *tool\_version*, *id=None*, *step\_id=None*, *resources=None*, *node\_group=None*)

Create a Node in this Network. The Node will be automatically added to the Network.

**Parameters**

- **tool** (`Union[Tool, str]`) – The Tool to base the Node on in the form: `name/space/toolname:version`
- **tool\_version** (`str`) – The version of the tool wrapper to use
- **id** (`Optional[str]`) – The id of the node to be created
- **step\_id** (`Optional[str]`) – The step to add the created node to
- **resources** (`Optional[ResourceLimit]`) – The resources required to run this node
- **node\_group** (`Optional[str]`) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.

**Return type**

`Node`

**Returns**

the newly created node

**create\_sink**(*datatype*, *id=None*, *step\_id=None*, *resources=None*, *node\_group=None*)

Create a SinkNode in this Network. The Node will be automatically added to the Network.

**Parameters**

- **datatype** (`Union[BaseDataType, str]`) – The DataType of the sink node
- **id** (`Optional[str]`) – The id of the sink node to be created
- **step\_id** (`Optional[str]`) – The step to add the created sink node to
- **resources** (`Optional[ResourceLimit]`) – The resources required to run this node
- **node\_group** (`str`) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.

**Return type**

`Node`

**Returns**

the newly created sink node

**create\_source**(*datatype*, *id=None*, *step\_id=None*, *resources=None*, *node\_group=None*)

Create a SourceNode in this Network. The Node will be automatically added to the Network.

**Parameters**

- **datatype** (`BaseDataType`) – The DataType of the source source\_node
- **id** (`str`) – The id of the source source\_node to be created
- **step\_id** (`str`) – The step to add the created source source\_node to
- **resources** (`Optional[ResourceLimit]`) – The resources required to run this node
- **node\_group** (`str`) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.

**Returns**

the newly created source source\_node

**Return type**

`SourceNode`

**draw**(*file\_path=None*, *draw\_dimensions=True*, *hide\_unconnected=True*, *expand\_macros=1*, *font\_size=14*)

Draw a graphical representation of the Network

**Parameters**

- **file\_path** (`str`) – The path of the file to create, the extension will control the image type
- **draw\_dimensions** (`bool`) – Flag to control if the dimension sizes should be drawn in the figure, default is true
- **expand\_macros** (`bool`) – Flag to control if and how macro nodes should be expanded, by default 1 level is expanded

**Return type**

`Optional[str]`

**Returns**

path of the image created or None if failed

**execute**(*source\_data*, *sink\_data*, *tmpdir=None*, *timestamp=None*, *blocking=True*, *execution\_plugin=None*, *tracking\_id=None*)

Execute the network with the given source and sink data.

#### Parameters

- **source\_data** (`Dict[str, Union[List[Union[Tuple[Union[str, int, float], ...], str, int, float]], Mapping[str, Union[Tuple[Union[str, int, float], ...], str, int, float]], Tuple[Union[str, int, float], ...], str, int, float]]`) – Source data to use as an input
- **sink\_data** (`Union[str, Dict[str, str]]`) – Sink rules to use for determining the outputs
- **tmpdir** (`Optional[str]`) – The scratch directory to use for this network run, if an existing directory is given, fastr will try to resume a network run (see [Continuing a Network](#))
- **timestamp** (`Union[datetime, str, None]`) – The timestamp of the network run (useful for retrying or continuing previous runs)
- **blocking** (`bool`) – Flag to indicate if the execution should be blocking or launched in a background thread
- **execution\_plugin** (`Optional[str]`) – The execution plugin to use for this run
- **tracking\_id** (`Optional[str]`) – The tracking id for this run, to be able to track this run from the logs

#### Return type

*NetworkRun*

#### Returns

The network run object for the started execution

**property id:** *str*

The unique id describing this resource

#### Return type

*str*

**classmethod load**(*filename*)

Load Network from a YAML file

#### Parameters

**filename** (*str*) –

#### Returns

loaded network

#### Return type

*Network*

**save**(*filename*)

Save the Network to a YAML file

#### Parameters

**filename** (`Union[str, Path]`) – Path of the file to save to

**property version:** *Version*

Version of the Network (so users can keep track of their version)

#### Return type

*Version*

**class** `fastr.api.Link`(*parent*)

Representation of a link for editing the Network

**property** `collapse`: `Tuple[Union[int, str], ...]`

The dimensions which the link will collapse into the cardinality

**Return type**

`Tuple[Union[int, str], ...]`

**property** `expand`: `bool`

Flag that indicates if the Link will expand the cardinality into a new dimension.

**Return type**

`bool`

**property** `id`: `str`

The unique id describing this resource

**Return type**

`str`

**class** `fastr.api.Node`(*parent*)

Representation of Node for editing the Network

**property** `id`: `str`

The unique id describing this resource

**Return type**

`str`

**property** `input`: `Input`

In case there is only a single Inputs in a Node, this can be used as a short hand. In that case it is basically the same as `list(node.inputs.values())[0]`.

**Return type**

`Input`

**property** `inputs`: `InputMap`

Mapping object containing all Inputs of a Node

**Return type**

`InputMap`

**property** `output`: `Output`

In case there is only a single Outputs in a Node, this can be used as a short hand. In that case it is basically the same as `list(node.outputs.values())[0]`.

**Return type**

`Output`

**property** `outputs`: `SubObjectMap[Output]`

Mapping object containing all Outputs of a Node

**Return type**

`SubObjectMap[Output]`

**class** `fastr.api.Input`(*parent*)

Representation of an Input of a Node

**\_\_lshift\_\_(other)**

This operator allows the easy creation of Links to this Input using the << operator. Creating links can be done by:

```
# Generic form
>> link = input << output
>> link = input << ['some', 'data'] # Create a constant node

# Examples
>> link1 = addint.inputs['left_hand'] << source1.input
>> link2 = addint.inputs['right_hand'] << [1, 2, 3]

# Multiple links
>> links = addints.inputs['left_hand'] << (source1.output, source2.output,
↪ source3.output)
```

The last example would return a tuple with three links.

**Parameters**

**other** (`Union[Output, BaseOutput, list, dict, tuple]`) – the target to create the link from, this can be an Output, a tuple of Outputs, or a data structure that can be used as the data for a ConstantNode

**Return type**

`Union[Link, Tuple[Link, ...]]`

**Returns**

Newly created link(s)

**\_\_rrshift\_\_(other)**

This operator allows to use the >> operator as alternative to using the << operator. See the [\\_\\_lshift\\_\\_ operator](#) for details.

**Parameters**

**other** (`Union[Output, BaseOutput, list, dict, tuple]`) – the target to create the link from

**Return type**

`Union[Link, Tuple[Link, ...]]`

**Returns**

Newly created link(s)

**append(value)**

Create a link from give resource to a new SubInput.

**Parameters**

**value** (`Union[Output, BaseOutput, list, dict, tuple]`) – The source for the link to be created

**Return type**

`Link`

**Returns**

The newly created link

**property id: str**

The unique id describing this resource

**Return type**`str`**property input\_group:** `str`

The input group of this Input. This property can be read and changed. Changing the input group of an Input will influence the data flow in a Node (see *Advanced flows in a Node* for details).

**Return type**`str`**class** `fastr.api.Output`(*parent*)

Representation of an Output of a Node

**\_\_getitem\_\_**(*item*)

Get a SubOutput of this Output. The SubOutput selects some data from the parent Output based on an index or slice of the cardinality.

**Parameters****item** (`Union[int, slice]`) – the key of the requested item, can be an index or slice**Return type**`Output`**Returns**

the requested SubOutput with a view of the data in this Output

**property id:** `str`

The unique id describing this resource

**Return type**`str`



## FASTR DEVELOPER MODULE REFERENCE

### 3.1 fastr Package

#### 3.1.1 fastr Package

Initialize self. See `help(type(self))` for accurate signature.

`fastr.__init__.__dir__()` → `list`  
default `dir()` implementation

`fastr.__init__.__format__()`  
default object formatter

`fastr.__init__.__init_subclass__()`  
This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

`fastr.__init__.__new__(*args, **kwargs)`  
Create and return a new object. See `help(type)` for accurate signature.

`fastr.__init__.__reduce__()`  
helper for pickle

`fastr.__init__.__reduce_ex__()`  
helper for pickle

`fastr.__init__.__sizeof__()` → `int`  
size of object in memory, in bytes

`fastr.__init__.__subclasshook__()`  
Abstract classes can override this to customize `issubclass()`.

This is invoked early on by `abc.ABCMeta.__subclasscheck__()`. It should return `True`, `False` or `NotImplemented`. If it returns `NotImplemented`, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

### 3.1.2 exceptions Module

This module contains all Fastr-related Exceptions

**exception** `fastr.exceptions.FastrAttributeError(*args, **kwargs)`

Bases: `FastrError`, `AttributeError`

AttributeError in the fastr system

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrCannotChangeAttributeError(*args, **kwargs)`

Bases: `FastrError`

Attempting to change an attribute of an object that can be set only once.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrCardinalityError(*args, **kwargs)`

Bases: `FastrError`

The description of the cardinality is not valid.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrCollectorError(*args, **kwargs)`

Bases: `FastrError`

Cannot collect the results from a Job because of an error

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrDataTypeFileNotReadable(*args, **kwargs)`

Bases: `FastrError`

Could not read the datatype file.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrDataTypeMismatchError(*args, **kwargs)`

Bases: `FastrError`

When using a DataType as the key for the DataTypeManager, the DataTypeManager found another DataType with the same name already in the DataTypeManager. The means fastr has two version of the same DataType in the system, which should never happen!

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrDataTypeNotAvailableError(*args, **kwargs)`

Bases: `FastrError`

The DataType requested is not found by the fastr system. Typically this means that no matching DataType is found in the DataTypeManager.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrDataTypeNotInstantiableError(*args, **kwargs)`

Bases: `FastrError`

The base classes for DataTypes cannot be instantiated and should always be sub-classed.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrDataTypeValueError(*args, **kwargs)
```

Bases: [FastrError](#)

This value in fastr did not pass the validation specified for its DataType, typically means that the data is missing or corrupt.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrError(*args, **kwargs)
```

Bases: [Exception](#)

This is the base class for all fastr related exceptions. Catching this class of exceptions should ensure a proper execution of fastr.

```
__init__(*args, **kwargs)
```

Constructor for all exceptions. Saves the caller object fullid (if found) and the file, function and line number where the object was created.

```
__module__ = 'fastr.exceptions'
```

```
__repr__()
```

String representation of the error

**Returns**

error representation

**Return type**

[str](#)

```
__str__()
```

String value of the error

**Returns**

error string

**Return type**

[str](#)

```
__weakref__
```

list of weak references to the object (if defined)

```
excerpt()
```

Return a excerpt of the Error as a tuple.

```
exception fastr.exceptions.FastrErrorInSubprocess(*args, **kwargs)
```

Bases: [FastrExecutionError](#)

Encountered an error in the subprocess started by the execution script

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrExecutableNotFoundError(executable=None, *args, **kwargs)
```

Bases: [FastrExecutionError](#)

The executable could not be found!

```
__init__(executable=None, *args, **kwargs)
```

Constructor for all exceptions. Saves the caller object fullid (if found) and the file, function and line number where the object was created.

```
__module__ = 'fastr.exceptions'
```

```
__str__()
```

String representation of the error

```
exception fastr.exceptions.FastrExecutionError(*args, **kwargs)
```

Bases: [FastrError](#)

Base class for all fastr execution related errors

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrFileNotFound(filepath, message=None)
```

Bases: [FastrError](#)

Could not find an expected file

```
__init__(filepath, message=None)
```

Constructor for all exceptions. Saves the caller object fullid (if found) and the file, function and line number where the object was created.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrIOError(*args, **kwargs)
```

Bases: [FastrError](#), [OSError](#)

IOError in the fastr system

```
__module__ = 'fastr.exceptions'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
exception fastr.exceptions.FastrImportError(*args, **kwargs)
```

Bases: [FastrError](#), [ImportError](#)

ImportError in the fastr system

```
__module__ = 'fastr.exceptions'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
exception fastr.exceptions.FastrIndexError(*args, **kwargs)
```

Bases: [FastrError](#), [IndexError](#)

IndexError in the fastr system

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrIndexNonexistent(*args, **kwargs)
```

Bases: [FastrIndexError](#)

This is an IndexError for samples requested from a sparse data array. The sample is not there but is probably not there because of sparseness rather than being a missing sample (e.g. out of bounds).

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrKeyError(*args, **kwargs)
```

Bases: [FastrError](#), [KeyError](#)

KeyError in the fastr system

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrLockNotAcquired(directory, message=None)
```

Bases: [FastrError](#)

Could not lock a directory

```
__init__(directory, message=None)
```

Constructor for all exceptions. Saves the caller object fullid (if found) and the file, function and line number where the object was created.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrLookupError(*args, **kwargs)
```

Bases: [FastrError](#)

Could not find specified object in the fastr environment.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrMountUnknownError(*args, **kwargs)
```

Bases: [FastrKeyError](#)

Trying to access an undefined mount

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNetworkMismatchError(*args, **kwargs)
```

Bases: [FastrError](#)

Two interacting objects belong to different fastr network.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNetworkUnknownError(*args, **kwargs)
```

Bases: [FastrKeyError](#)

Reference to a Tool that is not recognised by the fastr system. This typically means the specific id/version combination of the requested tool has not been loaded by the ToolManager.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNoValidTargetError(*args, **kwargs)
```

Bases: [FastrKeyError](#)

Cannot find a valid target for the tool

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNodeAlreadyPreparedError(*args, **kwargs)
```

Bases: [FastrStateError](#)

A attempt is made at preparing a NodeRun for the second time. This is not allowed as it would wipe the current execution data and cause data-loss.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNodeNotPreparedError(*args, **kwargs)
```

Bases: [FastrStateError](#)

When trying to access execution data of a NodeRun, the NodeRun must be prepared. The NodeRun has not been prepared by the execution, so the data is not available!

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNodeNotValidError(*args, **kwargs)
```

Bases: *FastrStateError*

A NodeRun is not in a valid state where it should be, typically an invalid NodeRun is passed to the executor causing trouble.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNotExecutableError(*args, **kwargs)
```

Bases: *FastrExecutionError*

The command invoked by subprocess is not executable on the system

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrNotImplementedError(*args, **kwargs)
```

Bases: *FastrError*, *NotImplementedError*

This function/method has not been implemented on purpose (e.g. should be overwritten in a sub-class)

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrOSError(*args, **kwargs)
```

Bases: *FastrError*, *OSError*

OSError in the fastr system

```
__module__ = 'fastr.exceptions'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
exception fastr.exceptions.FastrObjectUnknownError(*args, **kwargs)
```

Bases: *FastrKeyError*

Reference to a Tool that is not recognised by the fastr system. This typically means the specific id/version combination of the requested tool has not been loaded by the ToolManager.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrOptionalModuleNotAvailableError(*args, **kwargs)
```

Bases: *FastrNotImplementedError*

A optional modules for Fastr is needed for this function, but is not available on the current python installation.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrOutputValidationError(*args, **kwargs)
```

Bases: *FastrExecutionError*

An output of a Job does not pass validation

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrParentMismatchError(*args, **kwargs)
```

Bases: *FastrError*

Two interactive objects have different parent where they should be the same

```
__module__ = 'fastr.exceptions'
```

**exception** `fastr.exceptions.FastrPluginCapabilityNotImplemented(*args, **kwargs)`

Bases: `FastrNotImplementedError`

A plugin did not implement a capability that it advertised.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrPluginNotAvailable(*args, **kwargs)`

Bases: `FastrKeyError`

Indicates that a requested Plugin was not found on the system.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrPluginNotLoaded(*args, **kwargs)`

Bases: `FastrStateError`

The plugin was not successfully loaded. This means the plugin class cannot be instantiated.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrResultFileNotFound(filepath, message=None)`

Bases: `FastrFileNotFound`, `FastrExecutionError`

Could not found the result file of job that finished. This means the executionscript process was killed during interruption. Generally this means a scheduler killed it because of resource shortage.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrScriptNotFoundError(interpreter=None, script=None, paths=None, *args, **kwargs)`

Bases: `FastrExecutionError`

Script could not be found

`__init__`(*interpreter=None, script=None, paths=None, \*args, \*\*kwargs*)

Constructor for all exceptions. Saves the caller object fullid (if found) and the file, function and line number where the object was created.

`__module__` = `'fastr.exceptions'`

`__str__`()

String value of the error

#### Returns

error string

#### Return type

str

**exception** `fastr.exceptions.FastrSerializationError(message, serializer, original_exception=None)`

Bases: `FastrError`

The serialization encountered a serious problem

`__init__`(*message, serializer, original\_exception=None*)

Constructor for all exceptions. Saves the caller object fullid (if found) and the file, function and line number where the object was created.

`__module__` = `'fastr.exceptions'`

`__repr__()`

Simple string representation of the exception

`__str__()`

Advanced string representation of the exception including the data about where in the schema things went wrong.

**exception** `fastr.exceptions.FastrSerializationIgnoreDefaultError`(*message, serializer, original\_exception=None*)

Bases: [\*FastrSerializationError\*](#)

The value and default are both None, so the value should not be serialized.

`__module__ = 'fastr.exceptions'`

**exception** `fastr.exceptions.FastrSerializationInvalidDataError`(*message, serializer, original\_exception=None*)

Bases: [\*FastrSerializationError\*](#)

Encountered data to serialize that is invalid given the serialization schema.

`__module__ = 'fastr.exceptions'`

**exception** `fastr.exceptions.FastrSerializationMethodError`(\*args, \*\*kwargs)

Bases: [\*FastrKeyError\*](#)

The desired serialization method does not exist.

`__module__ = 'fastr.exceptions'`

**exception** `fastr.exceptions.FastrSinkDataUnavailableError`(\*args, \*\*kwargs)

Bases: [\*FastrKeyError\*](#)

Could not find the Sink data for the desire sink.

`__module__ = 'fastr.exceptions'`

**exception** `fastr.exceptions.FastrSizeInvalidError`(\*args, \*\*kwargs)

Bases: [\*FastrError\*](#)

The given size cannot be valid.

`__module__ = 'fastr.exceptions'`

**exception** `fastr.exceptions.FastrSizeMismatchError`(\*args, \*\*kwargs)

Bases: [\*FastrError\*](#)

The size of two object in fastr is not matching where it should.

`__module__ = 'fastr.exceptions'`

**exception** `fastr.exceptions.FastrSizeUnknownError`(\*args, \*\*kwargs)

Bases: [\*FastrError\*](#)

The size of object is not (yet) known and only a theoretical estimate is available at the moment.

`__module__ = 'fastr.exceptions'`

**exception** `fastr.exceptions.FastrSourceDataUnavailableError`(\*args, \*\*kwargs)

Bases: [\*FastrKeyError\*](#)

Could not find the Source data for the desire source.



```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrStateError(*args, **kwargs)
```

Bases: [FastrError](#)

An object is in an invalid/unexpected state.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrSubprocessNotFinished(*args, **kwargs)
```

Bases: [FastrExecutionError](#)

Encountered an error before the subprocess call by the execution script was properly finished.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrToolNotAvailableError(*args, **kwargs)
```

Bases: [FastrError](#)

The tool used is not available on the current platform (OS and architecture combination) and cannot be used.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrToolTargetNotFound(*args, **kwargs)
```

Bases: [FastrError](#)

Could not determine the location of the tools target binary/script. The tool cannot be used.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrToolUnknownError(*args, **kwargs)
```

Bases: [FastrKeyError](#)

Reference to a Tool that is not recognised by the fastr system. This typically means the specific id/version combination of the requested tool has not been loaded by the ToolManager.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrToolVersionError(*args, **kwargs)
```

Bases: [FastrError](#)

Version mismatch, usually the installed tool version and version requested by the network mismatch.

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrTypeError(*args, **kwargs)
```

Bases: [FastrError](#), [TypeError](#)

TypeError in the fastr system

```
__module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrUnknownURLSchemeError(*args, **kwargs)
```

Bases: [FastrKeyError](#)

Fastr encountered a data URL with a scheme that was not recognised by the IOPlugin manager.

```
__module__ = 'fastr.exceptions'
```

**exception** `fastr.exceptions.FastrValueError(*args, **kwargs)`

Bases: `FastrError`, `ValueError`

ValueError in the fastr system

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrVersionInvalidError(*args, **kwargs)`

Bases: `FastrValueError`

The string representation of the version is malformed.

`__module__` = `'fastr.exceptions'`

**exception** `fastr.exceptions.FastrVersionMismatchError(*args, **kwargs)`

Bases: `FastrValueError`

There is a mismatch between different parts of the Fastr environment and integrity is compromised.

`__module__` = `'fastr.exceptions'`

`fastr.exceptions.get_message(exception)`

Extract the message from an exception in a safe manner

**Parameters**

**exception** (`BaseException`) – exception to extract from

**Returns**

message string

**Return type**

`str`

### 3.1.3 globals Module

`fastr.globals.get_current_run()`

**Return type**

`Optional[NetworkRun]`

`fastr.globals.set_current_run(current_run)`

### 3.1.4 version Module

This module keeps track of the version of the currently used Fastr framework. It can check its version from mercurial or a saved file

`fastr.version.clear_version()`

Remove the cached version info

`fastr.version.get_base_version()`

Get the version from the top-level version file

**Return type**

`Optional[str]`

**Returns**

the version

**Rtype str**

`fastr.version.get_git_info()`

**Return type**

`Tuple[Optional[str], Optional[str]]`

`fastr.version.get_saved_version()`

Get cached version from file

**Return type**

`Tuple[Optional[str], Optional[str], Optional[str]]`

**Returns**

tuple with version, head revision and branch

`fastr.version.save_version(current_version, current_hg_head, current_hg_branch)`

Cache the version information (useful for when installing)

**Parameters**

- **current\_version** (*str*) – version
- **current\_hg\_head** (*str*) – mercurial head revision
- **current\_hg\_branch** (*str*) – mercurial branch

**Returns**

### 3.1.5 Subpackages

#### api Package

#### api Package

This module provides the API for fastr that users should use. This API will be considered stable between major versions. If users only interact via this API (and refrain from operating on parent attributes), their code should be compatible within major version of fastr.

**class** `fastr.api.ResourceLimit`(*cores=1, memory='2G', time=None, gpus=None*)

Bases: `object`

`__eq__(other)`

Check if two resource limits are equal

**Parameters**

**other** – resource limit to test against

**Return type**

`bool`

`__getstate__()`

**Return type**

`dict`

`__hash__ = None`

**\_\_init\_\_**(*cores=1, memory='2G', time=None, gpus=None*)

An object describing resource requirements/limits for a node

**Parameters**

- **cores** (`Optional[int]`) – number of cores
- **memory** (`Union[str, int, None]`) – memory specification, can be int with number of megabytes or a string with numbers ending on M, G, T, P for megabytes, gigabytes, terabytes or petabytes. Note that the number has to be an integer, e.g. 1500M would work, whereas 1.5G would be invalid
- **time** (`Union[str, int, None]`) – run time specification, this can be an int with the number of seconds or a string in the HH:MM:SS, MM:SS, or SS format. Where HH, MM, and SS are integers representing the number of hours, minutes and seconds.
- **gpus** (`Optional[int]`) – number of GPUs

**\_\_module\_\_** = 'fastr.core.resourcelimit'

**\_\_ne\_\_**(*other*)

Check if two resource limits are not equal

**Parameters**

**other** – resource limit to test against

**Return type**

`bool`

**\_\_setstate\_\_**(*state*)

**\_\_slots\_\_** = ('\_cores', '\_memory', '\_time', '\_gpus')

**copy**()

Return a copy of current resource limit object

**Return type**

`ResourceLimit`

**property cores:** `Optional[int]`

The required number of gpus

**Return type**

`Optional[int]`

**property gpus:** `Optional[int]`

The required number of gpus

**Return type**

`Optional[int]`

**property memory:** `Optional[int]`

The required memory in megabytes

**Return type**

`Optional[int]`

**classmethod set\_memory\_multiplier**(*value=typing.Union[int, NoneType]*)

**Return type**

`None`

**property time:** `int`

The required time in seconds

**Return type**

`int`

`fastr.api.create_network(id, version=None)`

Create a new Network object

**Parameters**

- **id** (`str`) – id of the network
- **version** (`Union[Version, str, None]`) – version of the network

**Return type**

`Network`

**Returns**

`fastr.api.create_network_copy(network_state)`

Create a network based on another Network state. The network state can be a Network or the state gotten from a Network with `__getstate__`.

**Parameters**

**network\_state** (`Union[Network, Network, dict]`) – Network (state) to create a copy of

**Return type**

`Network`

**Returns**

The rebuilt network

## core Package

## core Package

This module contains all of the core components of fastr. It has the classes to create networks and work with them.

## cardinality Module

`class fastr.core.cardinality.AnyCardinalitySpec(parent)`

Bases: `CardinalitySpec`

`__eq__(other)`

Test for equality

`__hash__ = None`

`__module__ = 'fastr.core.cardinality'`

`__str__()`

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

```
class fastr.core.cardinality.AsCardinalitySpec(parent, target)
```

Bases: `CardinalitySpec`

```
__eq__(other)
```

Test for equality

```
__hash__ = None
```

```
__init__(parent, target)
```

```
__module__ = 'fastr.core.cardinality'
```

```
__str__()
```

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

```
calculate_execution_cardinality(key=None)
```

Calculate the cardinality given the node and spec, during execution this should be available and not give unknowns once the data is present and the key is given.

**Parameters**

**key** – Key for which the cardinality is calculated

**Return type**

`Optional[int]`

**Returns**

calculated cardinality

```
calculate_job_cardinality(payload)
```

Calculate the actually cardinality when a job needs to know how many arguments to create for a non-automatic output.

**Return type**

`Optional[int]`

```
calculate_planning_cardinality()
```

Calculate the cardinality given the node and spec, for cardinalities that only have validation and not a pre-calculable value, this return None. :rtype: `Optional[int]` :return: calculated cardinality

```
get_orderdict_cardinality()
```

```
get_target()
```

**Return type**

`str`

**property node**

**property predefined**

Indicate whether the cardinality is predefined or can only be calculated after execution

```
class fastr.core.cardinality.CardinalitySpec(parent)
```

Bases: `object`

```
__dict__ = mappingproxy({'__module__': 'fastr.core.cardinality', '__init__':
<function CardinalitySpec.__init__>, '__str__': <function CardinalitySpec.__str__>,
'__repr__': <function CardinalitySpec.__repr__>, '__eq__': <function
CardinalitySpec.__eq__>, '__ne__': <function CardinalitySpec.__ne__>, 'predefined':
<property object>, 'validate': <function CardinalitySpec.validate>, '_validate':
<function CardinalitySpec._validate>, 'calculate_planning_cardinality': <function
CardinalitySpec.calculate_planning_cardinality>, 'calculate_execution_cardinality':
<function CardinalitySpec.calculate_execution_cardinality>,
'calculate_job_cardinality': <function CardinalitySpec.calculate_job_cardinality>,
'__dict__': <attribute '__dict__' of 'CardinalitySpec' objects>, '__weakref__':
<attribute '__weakref__' of 'CardinalitySpec' objects>, '__doc__': None,
'__hash__': None, '__annotations__': {}})
```

**abstract** `__eq__`(*other*)

Test for equality

**Return type**

`bool`

`__hash__` = None

`__init__`(*parent*)

`__module__` = 'fastr.core.cardinality'

`__ne__`(*other*)

Return self!=value.

**Return type**

`bool`

`__repr__`()

Console representation of the cardinality spec

**Return type**

`str`

**abstract** `__str__`()

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

`__weakref__`

list of weak references to the object (if defined)

**calculate\_execution\_cardinality**(*key=None*)

Calculate the cardinality given the node and spec, during execution this should be available and not give unknowns once the data is present and the key is given.

**Parameters**

**key** – Key for which the cardinality is calculated

**Return type**

`Optional[int]`

**Returns**

calculated cardinality

**calculate\_job\_cardinality**(*payload*)

Calculate the actually cardinality when a job needs to know how many arguments to create for a non-automatic output.

**Return type**

`Optional[int]`

**calculate\_planning\_cardinality**()

Calculate the cardinality given the node and spec, for cardinalities that only have validation and not a pre-calculable value, this return None. :rtype: `Optional[int]` :return: calculated cardinality

**property predefined**

Indicate whether the cardinality is predefined or can only be calculated after execution

**validate**(*payload*, *cardinality*, *planning=True*)

Validate cardinality given a payload and cardinality

**Parameters**

- **payload** (`Optional[dict]`) – Payload of the corresponding job
- **cardinality** (`int`) – Cardinality to validate
- **planning** (`bool`) – Indicate whether the is for the planning phase or not

**Return type**

`bool`

**Returns**

Validity of the cardinality given the spec and payload

**class** `fastr.core.cardinality.ChoiceCardinalitySpec`(*parent*, *options*)

Bases: `CardinalitySpec`

**\_\_eq\_\_**(*other*)

Test for equality

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(*parent*, *options*)

**\_\_module\_\_** = `'fastr.core.cardinality'`

**\_\_str\_\_**()

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

**class** `fastr.core.cardinality.IntCardinalitySpec`(*parent*, *value*)

Bases: `CardinalitySpec`

**\_\_eq\_\_**(*other*)

Test for equality

**Return type**

`bool`

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(*parent*, *value*)



```
__module__ = 'fastr.core.cardinality'
```

```
__str__()
```

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

```
calculate_execution_cardinality(key=None)
```

Calculate the cardinality given the node and spec, during execution this should be available and not give unknowns once the data is present and the key is given.

**Parameters**

**key** – Key for which the cardinality is calculated

**Return type**

`int`

**Returns**

calculated cardinality

```
calculate_job_cardinality(payload)
```

Calculate the actual cardinality when a job needs to know how many arguments to create for a non-automatic output.

**Return type**

`Optional[int]`

```
calculate_planning_cardinality()
```

Calculate the cardinality given the node and spec, for cardinalities that only have validation and not a pre-calculable value, this return None. :rtype: `int` :return: calculated cardinality

**property predefined**

Indicate whether the cardinality is predefined or can only be calculated after execution

```
class fastr.core.cardinality.MaxCardinalitySpec(parent, value)
```

Bases: `CardinalitySpec`

```
__eq__(other)
```

Test for equality

```
__hash__ = None
```

```
__init__(parent, value)
```

```
__module__ = 'fastr.core.cardinality'
```

```
__str__()
```

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

```
class fastr.core.cardinality.MinCardinalitySpec(parent, value)
```

Bases: `CardinalitySpec`

```
__eq__(other)
```

Test for equality

```
__hash__ = None
```

```
__init__(parent, value)
```

```
__module__ = 'fastr.core.cardinality'
```

```
__str__()
```

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

```
class fastr.core.cardinality.RangeCardinalitySpec(parent, min, max)
```

Bases: `CardinalitySpec`

```
__eq__(other)
```

Test for equality

```
__hash__ = None
```

```
__init__(parent, min, max)
```

```
__module__ = 'fastr.core.cardinality'
```

```
__str__()
```

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

```
class fastr.core.cardinality.ValueCardinalitySpec(parent, target)
```

Bases: `CardinalitySpec`

```
__eq__(other)
```

Test for equality

```
__hash__ = None
```

```
__init__(parent, target)
```

```
__module__ = 'fastr.core.cardinality'
```

```
__str__()
```

String version of the cardinality spec, should be parseable by `create_cardinality`

**Return type**

`str`

```
calculate_execution_cardinality(key=None)
```

Calculate the cardinality given the node and spec, during execution this should be available and not give unknowns once the data is present and the key is given.

**Parameters**

**key** – Key for which the cardinality is calculated

**Return type**

`Optional[int]`

**Returns**

calculated cardinality

**calculate\_job\_cardinality**(*payload*)

Calculate the actually cardinality when a job needs to know how many arguments to create for a non-automatic output.

**Return type**

`Optional[int]`

**property node****fastr.core.cardinality.create\_cardinality**(*desc*, *parent*)

Create simplified description of the cardinality. This changes the string representation to a tuple that is easier to check at a later time.

**Parameters**

- **desc** (`str`) – the string version of the cardinality
- **parent** – the parent input or output to which this cardinality spec belongs

**Return type**

`CardinalitySpec`

**Returns**

the simplified cardinality description

**Raises**

`FastrCardinalityError` – if the Input/Output has an incorrect cardinality description.

The translation works with the following table:

cardinality string	cardinality spec	description
"*", any	<code>((any,))</code>	Any cardinality is allowed
"N"	<code>('int', N)</code>	A cardinality of N is required
"N-M"	<code>('range', N, M)</code>	A cardinality between N and M is required
"*-M"	<code>('max', M)</code>	A cardinality of maximal M is required
"N-*"	<code>('min', N)</code>	A cardinality of minimal N is required
"[M,N,...,0,P]"	<code>('choice', [M,N,...,0,P])</code>	The cardinality should one of the given options
"as:input_id"	<code>('as', 'input_id')</code>	The cardinality should match the cardinality of the given Input
"val:input_id"	<code>('val', 'input_id')</code>	The cardinality should match the value of the given Input

---

**Note:** The maximum, minimum and range are inclusive

---

## dimension Module

**class** `fastr.core.dimension.Dimension(name, size)`

Bases: `object`

A class representing a dimension. It contains the name and size of the dimension.

`__eq__(other)`

Dimension is the same if the name and size are the same

**Return type**

`bool`

`__hash__ = None`

`__init__(name, size)`

The constructor for the dimension.

**Parameters**

- **name** (`str`) – Name of the dimension
- **size** (`int` or `Symbol`) – Size fo the dimension

`__module__ = 'fastr.core.dimension'`

`__ne__(other)`

The not equal test is simply the inverse of the equal test

**Return type**

`bool`

`__repr__()`

String representation of a Dimension

**Return type**

`str`

`__slots__ = ('_name', '_size')`

`copy()`

Get a copy object of a Dimension

**Return type**

`Dimension`

**property name:** `str`

**Return type**

`str`

**property size:** `SizeType`

**Return type**

`~SizeType`

`update_size(value)`

**class** fastr.core.dimension.ForwardsDimensionsBases: *HasDimensions*

Class of objects that have dimensions not because they contain data with dimensions but forward them (optionally with changes via `combine_dimensions`)

**\_\_abstractmethods\_\_** = frozenset({'combine\_dimensions', 'source'})**\_\_module\_\_** = 'fastr.core.dimension'**abstract** `combine_dimensions(dimensions)`

Method to combine/manipulate the dimensions

**Parameters****dimensions** – the input dimensions from the source**Returns**

dimensions manipulated for this object

**Return type**

tuple of dimensions

**property** `dimensions`: Tuple[*Dimension*, ...]

The dimensions of the object based on the forwarding

**Return type**Tuple[*Dimension*, ...]**abstract** **property** `source`: *HasDimensions*

The source object from which the dimensions are forwarded

**Returns**

the object from which the dimensions are forwarded

**Return type***HasDimensions***class** fastr.core.dimension.HasDimensionsBases: *object*

A Mixin class for any object that has a notion of dimensions and size. It uses the dimension property to expose the dimension name and size.

**\_\_abstractmethods\_\_** = frozenset({'dimensions'})

```
__dict__ = mappingproxy({'__module__': 'fastr.core.dimension', '__doc__': '\n A
Mixin class for any object that has a notion of dimensions and size. It\n uses the
dimension property to expose the dimension name and size.\n ', 'dimensions':
<property object>, 'dimnames': <property object>, 'size': <property object>,
'ndims': <property object>, '__dict__': <attribute '__dict__' of 'HasDimensions'
objects>, '__weakref__': <attribute '__weakref__' of 'HasDimensions' objects>,
'__abstractmethods__': frozenset({'dimensions'})}, '_abc_registry':
<_weakrefset.WeakSet object>, '_abc_cache': <_weakrefset.WeakSet object>,
'_abc_negative_cache': <_weakrefset.WeakSet object>, '_abc_negative_cache_version':
59, '__annotations__': {}})
```

**\_\_module\_\_** = 'fastr.core.dimension'**\_\_weakref\_\_**

list of weak references to the object (if defined)

**abstract property dimensions:** `Tuple[Dimension, ...]`

The dimensions has to be implemented by any subclass. It has to provide a tuple of Dimensions.

**Returns**

dimensions

**Return type**

`tuple`

**property dimnames:** `Tuple[str]`

A tuple containing the dimension names of this object. All items of the tuple are of type str.

**Return type**

`Tuple[str]`

**property ndims:** `int`

The number of dimensions in this object

**Return type**

`int`

**property size:** `Tuple[SizeType]`

A tuple containing the size of this object. All items of the tuple are of type int or Symbol.

**Return type**

`Tuple[~SizeType]`

## interface Module

**A module that describes the interface of a Tool. It specifies how a set of**

input values will be translated to commands to be executed. This creates a generic interface to different ways of executing underlying software.

**class** `fastr.core.interface.InputSpec`(*id\_*, *cardinality*, *datatype*, *required=False*, *description=""*, *default=None*, *hidden=False*)

Bases: `InputSpec`

Class containing the information about an Input Specification, this is essentially a data class (but

```
__dict__ = mappingproxy({'__module__': 'fastr.core.interface', '__doc__': '\n\nClass containing the information about an Input Specification, this is\nessentially a data class (but\n', '__new__': <staticmethod object>, 'asdict': <function\nInputSpec.asdict>, '__dict__': <attribute '__dict__' of 'InputSpec' objects>,\n '__annotations__': {}})
```

```
__module__ = 'fastr.core.interface'
```

```
static __new__(cls, id_, cardinality, datatype, required=False, description='', default=None,\n              hidden=False)
```

Create new instance of InputSpec(id, cardinality, datatype, required, description, default, hidden)

```
asdict()
```

`fastr.core.interface.InputSpecBase`

alias of `InputSpec`

**class** fastr.core.interface.Interface

Bases: Plugin, Serializable

Abstract base class of all Interfaces. Defines the minimal requirements for all Interface implementations.

**\_\_abstractmethods\_\_** = frozenset({'\_\_getstate\_\_', '\_\_setstate\_\_', 'execute', 'expanding', 'inputs', 'outputs'})

**abstract** \_\_getstate\_\_()

Retrieve the state of the Interface

**Returns**

the state of the object

**Rtype** dict

**\_\_module\_\_** = 'fastr.core.interface'

**abstract** \_\_setstate\_\_(state)

Set the state of the Interface

**abstract** execute(target, payload)

Execute the interface given the a target and payload. The payload should have the form:

```
{
  'input': {
    'input_id_a': (value, value),
    'input_id_b': (value, value)
  },
  'output': {
    'output_id_a': (value, value),
    'output_id_b': (value, value)
  }
}
```

**Parameters**

- **target** – the target to call
- **payload** – the payload to use

**Returns**

the result of the execution

**Return type**

(tuple of) *InterfaceResult*

**abstract property** expanding

Indicates whether or not this Interface will result in multiple samples per run. If the flow is unaffected, this will be zero, if it is nonzero it means that number of dimension will be added to the sample array.

**abstract property** inputs

OrderedDict of Inputs connected to the Interface. The format should be {input\_id: InputSpec}.

**abstract property** outputs

OrderedDict of Output connected to the Interface. The format should be {output\_id: OutputSpec}.

```
classmethod test()
```

Test the plugin, interfaces do not need to be tested on import

```
class fastr.core.interface.InterfaceResult(result_data, target_result, payload, sample_index=None,
                                           sample_id=None, errors=None)
```

Bases: [object](#)

The class in which Interfaces should wrap their results to be picked up by fastr

```
__dict__ = mappingproxy({'__module__': 'fastr.core.interface', '__doc__': '\n The
class in which Interfaces should wrap their results to be picked up by fastr\n ',
'__init__': <function InterfaceResult.__init__>, '__dict__': <attribute '__dict__'
of 'InterfaceResult' objects>, '__weakref__': <attribute '__weakref__' of
'InterfaceResult' objects>, '__annotations__': {}})
```

```
__init__(result_data, target_result, payload, sample_index=None, sample_id=None, errors=None)
```

```
__module__ = 'fastr.core.interface'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
class fastr.core.interface.OutputSpec(id_, cardinality, datatype, automatic=True, required=False,
                                       description="", hidden=False)
```

Bases: [OutputSpec](#)

Class containing the information about an Output Specification, this is essentially a data class (but

```
__dict__ = mappingproxy({'__module__': 'fastr.core.interface', '__doc__': '\n
Class containing the information about an Output Specification, this is\n
essentially a data class (but\n ', '__new__': <staticmethod object>, 'asdict':
<function OutputSpec.asdict>, '__dict__': <attribute '__dict__' of 'OutputSpec'
objects>, '__annotations__': {}})
```

```
__module__ = 'fastr.core.interface'
```

```
static __new__(cls, id_, cardinality, datatype, automatic=True, required=False, description="",
              hidden=False)
```

Create new instance of OutputSpec(id, cardinality, datatype, automatic, required, description, hidden)

```
asdict()
```

```
fastr.core.interface.OutputSpecBase
```

alias of [OutputSpec](#)

## ioplugin Module

This module contains the manager class for IOPlugins and the base class for all IOPlugins

```
class fastr.core.ioplugin.IOPlugin
```

Bases: [Plugin](#)

[IOPlugins](#) are used for data import and export for the sources and sinks. The main use of the [IOPlugins](#) is during execution (see [Execution](#)). The [IOPlugins](#) can be accessed via `fastr.ioplugins`, but generally there should be no need for direct interaction with these objects. The use of is mainly via the URL used to specify source and sink data.



```
__abstractmethods__ = frozenset({'scheme'})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.core.ioplugin'
```

```
cleanup()
```

(abstract) Clean up the IOPlugin. This is to do things like closing files or connections. Will be called when the plugin is no longer required.

```
expand_url(url)
```

(abstract) Expand an URL. This allows a source to collect multiple samples from a single url. The URL will have a wildcard or point to something with info and multiple urls will be returned.

**Parameters**

**url** (*str*) – url to expand

**Returns**

the resulting url(s), a tuple if multiple, otherwise a str

**Return type**

*str* or *tuple* of *str*

```
fetch_url(inurl, outfile)
```

(abstract) Fetch a file from an external data source.

**Parameters**

- **inurl** – url to the item in the data store
- **outpath** – path where to store the fetch data locally

```
fetch_value(inurl)
```

(abstract) Fetch a value from an external data source.

**Parameters**

**inurl** – the url of the value to retrieve

**Returns**

the fetched value

```
static isurl(string)
```

Test if given string is an url.

**Parameters**

**string** (*str*) – string to test

**Returns**

True if the string is an url, False otherwise

**Return type**

*bool*

```
path_to_url(path, mountpoint=None)
```

(abstract) Construct an url from a given mount point and a relative path to the mount point.

**Parameters**

- **path** (*str*) – the path to determine the url for

- **mountpoint** (*str* or *None*) – the mount point to use, will be automatically detected if *None* is given

**Returns**

url matching the path

**Return type**

*str*

**static print\_result(result)**

Print the result of the IOPlugin to stdout to be picked up by the tool

**Parameters**

**result** – value to print as a result

**Returns**

*None*

**pull\_source\_data(inurl, outdir, sample\_id, datatype=None)**

Transfer the source data from inurl to be available in outdir.

**Parameters**

- **inurl** (*str*) – the input url to fetch data from
- **outdir** (*str*) – the directory to write the data to
- **datatype** (*DataType*) – the datatype of the data, used for determining the total contents of the transfer

**Returns**

*None*

**push\_sink\_data(inpath, outurl, datatype=None)**

Write out the sink data from the inpath to the outurl.

**Parameters**

- **inpath** (*str*) – the path of the data to be pushed
- **outurl** (*str*) – the url to write the data to
- **datatype** (*DataType*) – the datatype of the data, used for determining the total contents of the transfer

**Returns**

*None*

**put\_url(inpath, outurl)**

(abstract) Put the files to the external data store.

**Parameters**

- **inpath** – path to the local data
- **outurl** – url to where to store the data in the external data store.

**put\_value(value, outurl)**

(abstract) Put the files to the external data store.

**Parameters**

- **value** – the value to store
- **outurl** – url to where to store the data in the external data store.

**abstract property scheme**

(abstract) This abstract property is to be overwritten by a subclass to indicate the url scheme associated with the IOPlugin.

**setup(\*args, \*\*kwargs)**

(abstract) Setup before data transfer. This can be any function that needs to be used to prepare the plugin for data transfer.

**url\_to\_path(url)**

(abstract) Get the path to a file from a url.

**Parameters**

**url** (*str*) – the url to retrieve the path for

**Returns**

the corresponding path

**Return type**

*str*

**provenance Module**

**class** fastr.core.provenance.**Provenance**(*host=None*)

Bases: *object*

The Provenance object keeps track of everything that happens to a data object.

```
__dict__ = mappingproxy({'__module__': 'fastr.core.provenance', '__doc__': '\n The
Provenance object keeps track of everything that happens to a data object.\n ',
'__init__': <function Provenance.__init__>, 'add_namespace': <function
Provenance.add_namespace>, 'agent': <function Provenance.agent>, 'activity':
<function Provenance.activity>, 'entity': <function Provenance.entity>,
'init_provenance': <function Provenance.init_provenance>, 'collect_provenance':
<function Provenance.collect_provenance>, 'collect_input_argument_provenance':
<function Provenance.collect_input_argument_provenance>, 'data_uri': <staticmethod
object>, 'get_parent_provenance': <staticmethod object>, 'serialize': <function
Provenance.serialize>, '__dict__': <attribute '__dict__' of 'Provenance' objects>,
'__weakref__': <attribute '__weakref__' of 'Provenance' objects>,
'__annotations__': {}})
```

```
__init__(host=None)
```

```
__module__ = 'fastr.core.provenance'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
activity(identifier, start_time=None, end_time=None, other_attributes=None)
```

```
agent(identifier, other_attributes=None)
```

```
collect_input_argument_provenance(input_argument)
```

```
collect_provenance(job, advanced_flow=False)
```

Collect the provenance for this job

```
static data_uri(value, job)
```

**entity**(*identifier*, *other\_attributes=None*)

**static get\_parent\_provenance**(*value*)

Find the provenance of the parent job

**Parameters**

**value** (*str*) – url for the value for which to find the job

**Returns**

the provenance of the job that created the value

**Raises**

- **FastrKeyError** – if the deferred is not available (yet)
- **FastrValueError** – if the value is not a valid deferred url

**init\_provenance**(*job*)

Create initial provenance document

**serialize**(*filename*, *format*)

## resourcelimit Module

Module for the management of resource limits of compute resources

**class** `fastr.core.resourcelimit.ResourceLimit`(*cores=1*, *memory='2G'*, *time=None*, *gpus=None*)

Bases: `object`

**\_\_annotations\_\_** = {}

**\_\_eq\_\_**(*other*)

Check if two resource limits are equal

**Parameters**

**other** – resource limit to test against

**Return type**

`bool`

**\_\_getstate\_\_**()

**Return type**

`dict`

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(*cores=1*, *memory='2G'*, *time=None*, *gpus=None*)

An object describing resource requirements/limits for a node

**Parameters**

- **cores** (`Optional[int]`) – number of cores
- **memory** (`Union[str, int, None]`) – memory specification, can be int with number of megabytes or a string with numbers ending on M, G, T, P for megabytes, gigabytes, terrabytes or petabytes. Note that the number has to be an integer, e.g. 1500M would work, whereas 1.5G would be invalid

- **time** (`Union[str, int, None]`) – run time specification, this can be an int with the number of seconds or a string in the HH:MM:SS, MM:SS, or SS format. Where HH, MM, and SS are integers representing the number of hours, minutes and seconds.
- **gpus** (`Optional[int]`) – number of GPUs

`__module__ = 'fastr.core.resourcelimit'`

`__ne__(other)`

Check if two resource limits are not equal

**Parameters**

**other** – resource limit to test against

**Return type**

`bool`

`__setstate__(state)`

`__slots__ = ('_cores', '_memory', '_time', '_gpus')`

`copy()`

Return a copy of current resource limit object

**Return type**

`ResourceLimit`

**property cores:** `Optional[int]`

The required number of gpus

**Return type**

`Optional[int]`

**property gpus:** `Optional[int]`

The required number of gpus

**Return type**

`Optional[int]`

**property memory:** `Optional[int]`

The required memory in megabytes

**Return type**

`Optional[int]`

**classmethod set\_memory\_multiplier**(`value=typing.Union[int, NoneType]`)

**Return type**

`None`

**property time:** `int`

The required time in seconds

**Return type**

`int`

## `samples` Module

This package holds the classes for working with samples.

**class** `fastr.core.samples.ContainsSamples`

Bases: *HasSamples*

`__abstractmethods__` = `frozenset({'samples'})`

`__getitem__`(*item*)

Return type

*SampleItem*

`__module__` = `'fastr.core.samples'`

`__setitem__`(*key*, *value*)

property `dimensions`: *Tuple[Dimension, ...]*

The dimensions has to be implemented by any subclass. It has to provide a tuple of Dimensions.

Returns

dimensions

Return type

*tuple*

abstract property `samples`: *SampleCollection*

Return type

*SampleCollection*

**class** `fastr.core.samples.HasSamples`

Bases: *HasDimensions*

Base class for all classes that supply samples. This base class allows to only define `__getitem__` and `size` and get all other basic functions mixed in so that the object behaves similar to a Mapping.

`__abstractmethods__` = `frozenset({'__getitem__', 'dimensions'})`

`__contains__`(*item*)

Return type

*bool*

abstract `__getitem__`(*item*)

Return type

*SampleItem*

`__iter__`()

Return type

*SampleIndex*

`__module__` = `'fastr.core.samples'`

`ids`()

Return type

*List[SampleId]*

**indexes()**

**Return type**

`List[SampleIndex]`

**items()**

**Return type**

`List[SampleItem]`

**iteritems()**

**Return type**

`SampleItem`

**class** `fastr.core.samples.SampleBaseId(*args: Union[ElementType, Iterable[ElementType]])`

Bases: `tuple`, `Generic[ElementType]`

This class represents a sample id. A sample id is a multi-dimensional id that has a simple, consistent string representation.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_add\_\_**(*other*)

Add another SampleId, this allows to add parts to the SampleId in a convenient way.

**Return type**

`SampleBaseId`

**\_\_annotations\_\_** = {'\_element\_type': `typing.ClassVar[typing.Type[~ElementType]]`}

**\_\_args\_\_** = `None`

**\_\_dict\_\_** = `mappingproxy({'__module__': 'fastr.core.samples', '__annotations__': {'_element_type': typing.ClassVar[typing.Type[~ElementType]]}, '__doc__': '\n This class represents a sample id. A sample id is a multi-dimensional\n id that has a simple, consistent string representation.\n ', '_element_type': None, '__new__': <staticmethod object>, '__getnewargs__': <function SampleBaseId.__getnewargs__>, '__repr__': <function SampleBaseId.__repr__>, '__str__': <function SampleBaseId.__str__>, '__add__': <function SampleBaseId.__add__>, '__radd__': <function SampleBaseId.__radd__>, '__origin__': None, '__extra__': None, '__gorg__': fastr.core.samples.SampleBaseId, '__dict__': <attribute \'__dict__\' of \'SampleBaseId\' objects>, '__abstractmethods__': frozenset(), '_abc_registry': <weakrefset.WeakSet object>, '_abc_cache': <weakrefset.WeakSet object>, '_abc_generic_negative_cache': <weakrefset.WeakSet object>, '_abc_generic_negative_cache_version': 59, '__parameters__': (~ElementType), '__args__': None, '__next_in_mro__': <class \'object\'>, '__orig_bases__': (<class \'tuple\'>, typing.Generic[~ElementType]), '__tree_hash__': -9223366124441449276})`

**\_\_extra\_\_** = `None`

**\_\_getnewargs\_\_**()

Get new args gives the arguments to use to re-create this object, This is used for serialization.

**Return type**

`Tuple[~ElementType, ...]`

**\_\_module\_\_** = `'fastr.core.samples'`

**static** `__new__(cls, *args)`

Create a new SampleId

**Parameters**

**args** (*iterator/iterable of element type or element type*) – the strings to make sample id for

`__next_in_mro__`

alias of `object`

`__orig_bases__` = (`<class 'tuple'>`, `typing.Generic[~ElementType]`)

`__origin__` = `None`

`__parameters__` = (`~ElementType`,)

`__radd__`(*other*)

Add another SampleId, this allows to add parts to the SampleId in a convenient way. This is the right-hand version of the operator.

**Return type**

`SampleBaseId`

`__repr__`()

Get a string representation for the SampleBaseId

**Returns**

the string representation

**Return type**

`str`

`__str__`()

Get a string version for the SampleId, joins the SampleId with `__` to create a single string version.

**Returns**

the string version

**Return type**

`str`

`__tree_hash__` = `-9223366124441449276`

**class** `fastr.core.samples.SampleCollection(dimnames, parent)`

Bases: `MutableMapping`, `HasDimensions`

The SampleCollections is a class that contains the data including a form of ordering. Each sample is reachable both by its SampleId and a SampleIndex. The object is sparse, so not all SampleId have to be defined allowing for non-rectangular data shapes.

---

**Note:** This object is meant to replace both the SampleIdList and the ValueStorage.

---

`__abstractmethods__` = `frozenset({})`

`__contains__`(*item*)

Check if an item is in the SampleCollection. The item can be a SampleId or SampleIndex. If the item is a slicing SampleIndex, then check if it would return any data (True) or no data (False)



**Parameters**

**item** (*SampleId*, *SampleIndex*) – the item to check for

**Returns**

flag indicating item is in the collections

**Return type**

*bool*

**`__delitem__`**(*key*)

Remove an item from the SampleCollection

**Parameters**

**key** (*SampleId*, *SampleIndex*, *tuple of both*, or *SampleItem*) – the key of the item to remove

```
__dict__ = mappingproxy({'__module__': 'fastr.core.samples', '__doc__': '\n The
SampleCollections is a class that contains the data including a form\n of ordering.
Each sample is reachable both by its SampleId and a\n SampleIndex. The object is
sparse, so not all SampleId have to be defined\n allowing for non-rectangular data
shapes.\n\n .. note::\n\n This object is meant to replace both the SampleIdList and
the\n ValueStorage.\n ', '__init__': <function SampleCollection.__init__>,
'__repr__': <function SampleCollection.__repr__>, '__contains__': <function
SampleCollection.__contains__>, '__getitem__': <function
SampleCollection.__getitem__>, '__setitem__': <function
SampleCollection.__setitem__>, '__delitem__': <function
SampleCollection.__delitem__>, '__iter__': <function SampleCollection.__iter__>,
'__len__': <function SampleCollection.__len__>, 'dimensions': <property object>,
'ndims': <property object>, 'parent': <property object>, 'fullid': <property
object>, '__dict__': <attribute '__dict__' of 'SampleCollection' objects>,
'__weakref__': <attribute '__weakref__' of 'SampleCollection' objects>,
'__abstractmethods__': frozenset(), '_abc_registry': <_weakrefset.WeakSet object>,
'_abc_cache': <_weakrefset.WeakSet object>, '_abc_negative_cache':
<_weakrefset.WeakSet object>, '_abc_negative_cache_version': 59, '__annotations__':
{}})
```

**`__getitem__`**(*item*)

Retrieve (a) SampleItem(s) from the SampleCollection using the SampleId or SampleIndex. If the item is a tuple, it should be valid tuple for constructing either a SampleId or SampleIndex.

**Parameters**

**item** (*SampleId*, *SampleIndex*, or *tuple*) – the identifier of the item to retrieve

**Returns**

the requested item

**Return type**

*SampleItem*

**Raises**

- **FastrTypeError** – if the item parameter is of incorrect type
- **KeyError** – if the item is not found

**`__init__`**(*dimnames*, *parent*)

Create a new SampleCollection

**`__iter__`**()

Iterate over the indices

**Return type***SampleIndex***\_\_len\_\_()**

Get the number of samples in the SampleCollections.

**Return type***int***\_\_module\_\_** = 'fastr.core.samples'**\_\_repr\_\_()**

Return repr(self).

**Return type***str***\_\_setitem\_\_(key, value)**

Set an item to the SampleCollection. The key can be a SampleId, SampleIndex or a tuple containing a SampleId and SampleIndex. The value can be a SampleItem (with the SampleId and SampleIndex matching), a tuple with values (assuming no depending jobs), or a with a list of values and a set of depending jobs.

**Parameters**

- **key** (*SampleId*, *SampleIndex*, *tuple of both*, or *SampleItem*) – the key of the item to store
- **value** (*SampleItem*, *tuple of values*, or *tuple of tuple of values and set of depending jobs*) – the value of the SampleItem to store

**Raises**

- *FastrTypeError* – if the key or value types are incorrect
- *FastrValueError* – if the id or values are incorrectly formed

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**property dimensions:** *Tuple[Dimension, ...]*

The dimensions has to be implemented by any subclass. It has to provide a tuple of Dimensions.

**Returns***dimensions***Return type***tuple***property fullid:** *str*

The full defining ID for the SampleIdList

**Return type***str***property ndims:** *int*

The number of dimensions in this SampleCollection

**Return type***int***property parent**

The parent object holding the SampleCollection

```
class fastr.core.samples.SampleId(*args: Union[ElementType, Iterable[ElementType]])
```

Bases: *SampleBaseId*

SampleId is an identifier for data using human readable strings

```
__abstractmethods__ = frozenset({})
```

```
__args__ = None
```

```
__extra__ = None
```

```
__module__ = 'fastr.core.samples'
```

```
__next_in_mro__
```

alias of *object*

```
__orig_bases__ = (fastr.core.samples.SampleBaseId,)
```

```
__origin__ = None
```

```
__parameters__ = ()
```

```
__tree_hash__ = -9223366124441449110
```

```
class fastr.core.samples.SampleIndex(*args: Union[ElementType, Iterable[ElementType]])
```

Bases: *SampleBaseId*

SampleId is an identifier for data using the location in the N-d data structure.

```
__abstractmethods__ = frozenset({})
```

```
__args__ = None
```

```
__extra__ = None
```

```
__module__ = 'fastr.core.samples'
```

```
__next_in_mro__
```

alias of *object*

```
__orig_bases__ = (fastr.core.samples.SampleBaseId,)
```

```
__origin__ = None
```

```
__parameters__ = ()
```

```
__repr__()
```

Get a string representation for the SampleIndex

**Returns**

the string representation

**Return type**

*str*

```
__str__()
```

Get a string version for the SampleId, joins the SampleId with \_\_ to create a single string version.

**Returns**

the string version

**Return type**

*str*

`__tree_hash__ = -9223366124441894413`

**expand**(*size*)

Function expanding a slice SampleIndex into a list of non-slice SampleIndex objects

**Parameters**

**size** (*Sequence*[*int*]) – the size of the collection to slice

**Return type**

*Tuple*[*SampleIndex*, ...]

**property isslice:** *bool*

Flag indicating that the SampleIndex is a slice (as opposed to a simple single index).

**Return type**

*bool*

**class** `fastr.core.samples.SampleItem`(*index*, *id*, *data*, *jobs*=None, *failed\_annotations*=None, *status*=*SampleState.VALID*)

Bases: *SampleItemBase*

`__module__ = 'fastr.core.samples'`

**static** `__new__`(*cls*, *index*, *id*, *data*, *jobs*=None, *failed\_annotations*=None, *status*=*SampleState.VALID*)

Create a SampleItem. Data should be an OrderedDict of tuples.

**Parameters**

- **index** (*tuple*, *slice*) – the sample index
- **id** (*SampleId*) – the sample id
- **data** (*SampleValue*, *Mapping*) – the data values
- **jobs** (*set*) – set of jobs on which this SampleItems data depends.
- **failed\_annotations** (*set*) – set of tuples. The tuple is constructed like follows: (job\_id, reason).

**class** `fastr.core.samples.SampleItemBase`(*index*, *id*, *data*, *jobs*=None, *failed\_annotations*=None, *status*=*SampleState.VALID*)

Bases: *tuple*

This class represents a sample item, a combination of a SampleIndex, SampleID, value and required jobs. The SampleItem based on a named tuple and has some extra methods to combine SampleItems easily.

`__add__`(*other*)

The addition operator combines two SampleItems into a single SampleItems. It merges the data and jobs and takes the index and id of the left-hand item.

**Parameters**

**other** (*SampleItem*) – The other item to add to this one

**Returns**

the combined SampleItem

**Return type**

*SampleItem*

```
__dict__ = mappingproxy({'__module__': 'fastr.core.samples', '__doc__': '\n This
class represents a sample item, a combination of a SampleIndex,\n SampleID, value
and required jobs. The SampleItem based on a named\n tuple and has some extra
methods to combine SampleItems easily.\n ', '__new__': <staticmethod object>,
'__repr__': <function SampleItemBase.__repr__>, '__getnewargs__': <function
SampleItemBase.__getnewargs__>, '__add__': <function SampleItemBase.__add__>,
'combine': <staticmethod object>, 'replace': <function SampleItemBase.replace>,
'index': <property object>, 'id': <property object>, 'data': <property object>,
'jobs': <property object>, 'failed_annotations': <property object>, 'status':
<property object>, 'cardinality': <property object>, 'dimensionality': <property
object>, '__dict__': <attribute '__dict__' of 'SampleItemBase' objects>,
'__annotations__': {}})
```

```
__getnewargs__()
```

Get new args gives the arguments to use to re-create this object, This is used for serialization.

**Return type**

`Tuple`

```
__module__ = 'fastr.core.samples'
```

```
static __new__(cls, index, id, data, jobs=None, failed_annotations=None, status=SampleState.VALID)
```

Create a SampleItem. Data should be an OrderedDict of tuples.

**Parameters**

- **index** (`tuple`, `slice`) – the sample index
- **id** (`SampleId`) – the sample id
- **data** (`SampleValue`, `Mapping`) – the data values
- **jobs** (`set`) – set, tuple or list of jobs on which this SampleItems data depends.
- **failed\_annotations** (`set`) – set of tuples. The tuple is constructed like follows: (job\_id, reason).

```
__repr__()
```

Get a string representation for the SampleItem

**Returns**

the string representation

**Return type**

`str`

```
property cardinality: int
```

The cardinality of this Sample

**Return type**

`int`

```
static combine(*args)
```

Combine a number of SampleItems into a new one.

**Parameters**

**args** (*iterable of SampleItems*) – the SampleItems to combine

**Returns**

the combined SampleItem

**Return type**

*SampleItem*

It is possible to both give multiple arguments, where each argument is a *SampleItem*, or a single argument which is an iterable yielding *SampleItems*.

```
# variables a, b, c, d are SampleItems to combine
# These are all valid ways of combining the SampleItems
comb1 = SampleItem.combine(a, b, c, d) # Using multiple arguments
l = [a, b, c, d]
comb2 = SampleItem.combine(l) # Using a list of arguments
comb3 = SampleItem.combine(l.__iter__()) # Using an iterator
```

**property data:** *SampleValue*

The data *SampleValue* of the *SampleItem*

**Returns**

The value of this *SampleItem*

**Return type**

*SampleValue*

**property dimensionality:** *int*

The dimensionality of this *Sample*

**Return type**

*int*

**property failed\_annotations**

**property id:** *SampleId*

The sample id of the *SampleItem*

**Returns**

The id of this *SampleItem*

**Return type**

*SampleId*

**property index:** *SampleIndex*

The index of the *SampleItem*

**Returns**

The index of this *SampleItem*

**Return type**

*SampleIndex*

**property jobs:** *Set*

The set of the jobs on which this *SampleItem* depends

**Returns**

The jobs that generated the data for this *SampleItem*

**Return type**

*set*

**replace**(*index=None, id=None, data=None, jobs=None, failed\_annotations=None, status=None*)

Create a new version of the objects with fields replaced

**Parameters**

- **index** – new index to use
- **id** – new id to use
- **data** – new data to use
- **jobs** – new jobs to use
- **failed\_annotations** – new failed annotations to use
- **status** – new status to use

**Returns**

new version of object with given fields replaced

**property status:** *SampleState*

**Return type**

*SampleState*

```
class fastr.core.samples.SamplePayload(index, id, data, jobs=None, failed_annotations=None,
                                       status=SampleState.VALID)
```

Bases: *SampleItemBase*

**\_\_add\_\_**(*other*)

The addition operator combines two SampleItems into a single SampleItems. It merges the data and jobs and takes the index and id of the left-hand item.

**Parameters**

**other** (*SampleItem*) – The other item to add to this one

**Returns**

the combined SamplePayload

**Return type**

*SamplePayload*

**\_\_module\_\_** = 'fastr.core.samples'

```
static __new__(cls, index, id, data, jobs=None, failed_annotations=None, status=SampleState.VALID)
```

Create a SamplePayload. Data should be an OrderedDict of tuples.

**Parameters**

- **index** (*tuple*, *slice*) – the sample index
- **id** (*SampleId*) – the sample id
- **data** (*SampleValue*, *Mapping*) – the data values
- **jobs** (*set*) – set of jobs on which this SampleItems data depends.
- **failed\_annotations** (*set*) – set of tuples. The tuple is constructed like follows: (job\_id, reason).

```
class fastr.core.samples.SampleState(value)
```

Bases: *Enum*

Possible states a SampleItem can be in. This is to annotate if data is missing from the start, or missing due to failure.

**FAILED** = 'FAILED'

**MISSING** = 'MISSING'

```
VALID = 'VALID'

__module__ = 'fastr.core.samples'

classmethod combine(states)

class fastr.core.samples.SampleValue(*args, **kwargs)
    Bases: MutableMapping
    A collection containing the content of a sample
    __abstractmethods__ = frozenset({})

    __add__(other)

        Return type
        SampleValue

    __annotations__ = {'_key_type': typing.ClassVar[typing.Tuple[typing.Type, ...]]}

    __delitem__(key)

    __dict__ = mappingproxy({'__module__': 'fastr.core.samples', '__annotations__':
{'_key_type': typing.ClassVar[typing.Tuple[typing.Type, ...]]}, '__doc__': '\n A
collection containing the content of a sample\n ', '_key_type': (<class 'int'>,
<class 'str'>), '__init__': <function SampleValue.__init__>, '__repr__': <function
SampleValue.__repr__>, '__getitem__': <function SampleValue.__getitem__>,
'__setitem__': <function SampleValue.__setitem__>, '__getstate__': <function
SampleValue.__getstate__>, '__setstate__': <function SampleValue.__setstate__>,
'__delitem__': <function SampleValue.__delitem__>, '__len__': <function
SampleValue.__len__>, '__iter__': <function SampleValue.__iter__>, 'is_sequence':
<property object>, 'is_mapping': <property object>, 'sequence_part': <function
SampleValue.sequence_part>, 'mapping_part': <function SampleValue.mapping_part>,
'cast': <function SampleValue.cast>, 'iterelements': <function
SampleValue.iterelements>, '__radd__': <function SampleValue.__radd__>, '__add__':
<function SampleValue.__add__>, '__dict__': <attribute '__dict__' of 'SampleValue'
objects>, '__weakref__': <attribute '__weakref__' of 'SampleValue' objects>,
'__abstractmethods__': frozenset(), '_abc_registry': <_weakrefset.WeakSet object>,
'_abc_cache': <_weakrefset.WeakSet object>, '_abc_negative_cache':
<_weakrefset.WeakSet object>, '_abc_negative_cache_version': 59})

    __getitem__(item)

    __getstate__()

    __init__(*args, **kwargs)

    __iter__()

        Return type
        Union[str, int]

    __len__()

        Return type
        int

    __module__ = 'fastr.core.samples'
```



```

__radd__(other)

    Return type
    SampleValue

__repr__()
    Return repr(self).

    Return type
    str

__setitem__(key, value)

__setstate__(state)

__weakref__
    list of weak references to the object (if defined)

cast(datatype)

property is_mapping: bool

    Return type
    bool

property is_sequence: bool

    Return type
    bool

iterelements()

mapping_part()

sequence_part()

```

## target Module

The module containing the classes describing the targets.

**class** `fastr.core.target.ProcessUsageCollection`

Bases: *Sequence*

```

__abstractmethods__ = frozenset({})

__dict__ = mappingproxy({'__module__': 'fastr.core.target', 'usage_type': <class
'fastr.core.target.SystemUsageInfo'>, '__init__': <function
ProcessUsageCollection.__init__>, '__len__': <function
ProcessUsageCollection.__len__>, '__getitem__': <function
ProcessUsageCollection.__getitem__>, 'append': <function
ProcessUsageCollection.append>, 'aggregate': <function
ProcessUsageCollection.aggregate>, '__dict__': <attribute '__dict__' of
'ProcessUsageCollection' objects>, '__weakref__': <attribute '__weakref__' of
'ProcessUsageCollection' objects>, '__doc__': None, '__abstractmethods__':
frozenset(), '_abc_registry': <_weakrefset.WeakSet object>, '_abc_cache':
<_weakrefset.WeakSet object>, '_abc_negative_cache': <_weakrefset.WeakSet object>,
'_abc_negative_cache_version': 59, '__annotations__': {}})

```

`__getitem__(item)`  
`__init__()`  
`__len__()`  
`__module__ = 'fastr.core.target'`  
`__weakref__`  
list of weak references to the object (if defined)  
`aggregate(number_of_points)`  
`append(value)`  
`usage_type`  
alias of [SystemUsageInfo](#)

**class** `fastr.core.target.SubprocessBasedTarget`

Bases: [Target](#)

Abstract based class for targets which call the target via a subprocess. Supplies a `call_subprocess` which executes the command and profiles the resulting subprocess.

`__abstractmethods__ = frozenset({'run_command'})`  
`__module__ = 'fastr.core.target'`

`call_subprocess(command)`

Call a subprocess with logging/timing/profiling

**Parameters**

**command** ([list](#)) – the command to execute

**Returns**

execution info

**Return type**

[dict](#)

`monitor_process(process, resources)`

Monitor a process and profile the cpu, memory and io use. Register the resource use every `_MONITOR_INTERVAL` seconds.

**Parameters**

- **process** ([subproces.Popen](#)) – process to monitor
- **resources** ([ProcessUsageCollection](#)) – list to append measurements to

**class** `fastr.core.target.SystemUsageInfo(timestamp, cpu_percent, vmem, rmem, read_bytes, write_bytes)`

Bases: [tuple](#)

`__getnewargs__()`

Return self as a plain tuple. Used by copy and pickle.

`__module__ = 'fastr.core.target'`

**static** `__new__(cls, timestamp, cpu_percent, vmem, rmem, read_bytes, write_bytes)`

Create new instance of `SystemUsageInfo(timestamp, cpu_percent, vmem, rmem, read_bytes, write_bytes)`

**\_\_repr\_\_()**

Return a nicely formatted representation string

**\_\_slots\_\_ = ()**

**property cpu\_percent**

Alias for field number 1

**property read\_bytes**

Alias for field number 4

**property rmem**

Alias for field number 3

**property timestamp**

Alias for field number 0

**property vmem**

Alias for field number 2

**property write\_bytes**

Alias for field number 5

**class fastr.core.target.Target**

Bases: Plugin

The abstract base class for all targets. Execution with a target should follow the following pattern:

```
>>> with Target() as target:
...     target.run_command(['sleep', '10'])
```

The Target context operator will set the correct paths/initialization. Within the context command can be ran and when leaving the context the target reverts the state before.

**\_\_abstractmethods\_\_ = frozenset({'run\_command'})**

**\_\_enter\_\_()**

Set the environment in such a way that the target will be on the path.

**\_\_exit\_\_(exc\_type, exc\_value, traceback)**

Cleanup the environment where needed

**\_\_module\_\_ = 'fastr.core.target'**

**abstract run\_command(command)**

Run a command with the target

**Return type**

*TargetResult*

**classmethod test()**

Test the plugin, interfaces do not need to be tested on import

**class fastr.core.target.TargetResult**(*return\_code, stdout, stderr, command, resource\_usage, time\_elapsed*)

Bases: *object*

```
__dict__ = mappingproxy({'__module__': 'fastr.core.target', '__init__': <function
TargetResult.__init__>, 'as_dict': <function TargetResult.as_dict>, '__dict__':
<attribute '__dict__' of 'TargetResult' objects>, '__weakref__': <attribute
'__weakref__' of 'TargetResult' objects>, '__doc__': None, '__annotations__': {}})
```

```
__init__(return_code, stdout, stderr, command, resource_usage, time_elapsed)
```

Class to formalize the resulting data of a Target

#### Parameters

- **return\_code** (`int`) – the return code of the process
- **stdout** (`Union[str, bytes]`) – the stdout generated by the process
- **stderr** (`Union[str, bytes]`) – the stderr generated by the process
- **command** (`List[Union[str, bytes]]`) – the command executed
- **resource\_usage** (`List[SystemUsageInfo]`) – the resource use during execution
- **time\_elapsed** (`int`) – time used (in seconds)

```
__module__ = 'fastr.core.target'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
as_dict()
```

A dictionary of the data in the object (meant for serialization)

#### Return type

`Dict[str, Union[int, str, List]]`

## tool Module

A module to maintain a tool.

Exported classes:

- **Tool** – A class encapsulating a tool.
- **ParameterDescription** – The base class containing the shared description of a parameter (both input and output).
- **InputParameterDescription** – A class containing the description of an input parameter.
- **OutputParameterDescription** – A class containing the description of an output parameter.

```
class fastr.core.tool.Tool(doc=None)
```

Bases: `Serializable`

The class encapsulating a tool.

```
DEFAULT_TARGET_CLASS = {'MacroNode': 'MacroTarget'}
```

```
TOOL_REFERENCE_FILE_NAME = '__fastr_tool_ref__.json'
```

```
TOOL_RESULT_FILE_NAME = '__fastr_tool_result.pickle.gz'
```

```
__dataschemafile__ = 'Tool.schema.json'
```

**\_\_eq\_\_**(*other*)

Compare two Tool instances with each other.

**Parameters**

**other** ([Tool](#)) – the other instances to compare to

**Returns**

True if equal, False otherwise

**\_\_getstate\_\_**()

Retrieve the state of the Tool

**Returns**

the state of the object

**Rtype** dict

**\_\_hash\_\_** = None

**\_\_init\_\_**(*doc=None*)

Create a new Tool :param doc: path of toolfile or a dict containing the tool data :type doc: str or dict

**\_\_module\_\_** = 'fastr.core.tool'

**\_\_repr\_\_**()

Get a string representation for the Tool. This will show the inputs and output defined in a table-like structure.

**Returns**

the string representation

**Return type**

str

**\_\_setstate\_\_**(*state*)

Set the state of the Tool by the given state.

**Parameters**

**state** ([dict](#)) – The state to populate the object with

**\_\_str\_\_**()

Get a string version for the Tool

**Returns**

the string version

**Return type**

str

**authors**

List of authors of the tool. These people wrapped the executable but are not responsible for executable itself.

**cite**

This holds the citation you should use when publishing something based on this Tool

**command**

Command is a dictionary contain information about the command which is called by this Tool: command['interpreter'] holds the (possible) interpreter to use command['targets'] holds a per os/arch dictionary of files that should be executed command['url'] is the webpage of the command to be called command['version'] is the version of the command used command['description'] can help a description of the command command['authors'] lists the original authors of the command

**property command\_version**

**static compare\_output\_data**(*current\_output\_data*, *reference\_output\_data*, *validation\_result*, *output*)

**create\_reference**(*input\_data*, *output\_directory*, *mount\_name*='\_\_ref\_tmp\_\_', *copy\_input*=True)

**description**

Description of the tool and it's functionality

**execute**(*payload*=None, *\*\*kwargs*)

Execute a Tool given the payload for a single run

**Parameters**

**payload** – the data to execute the Tool with

**Returns**

The result of the execution

**Return type**

InterFaceResult

**property fullid**

The full id of this tool

**property hash**

**help**

Man page for the Tool. Here usage and examples can be described in detail

**property id**

**property inputs**

**name**

Name of the tool, this should be a descriptive, human readable name.

**namespace**

The namespace this tools lives in, this will be set by the ToolManager on load

**node\_class**

Class for of the Node to use

**property ns\_id**

The namespace and id of the Tool

**property outputs**

**property path**

The path of the directory in which the tool definition file was located.

**references**

A list of documents and in depth reading about the methods used in this tool

**requirements**

Requirements for this Tool

<b>Warning:</b> Not yet implemented
-------------------------------------

**serialize()**

Prepare data for serialization, this removes some fields from the state that are not needed when serializing to a file

**tags**

List of tags for this tool

**property target**

The OS and arch matched target definition.

**test(reference=None)**

Run the tests for this tool

**test\_spec**

alias of TestSpecification

**classmethod test\_tool(reference\_data\_dir, tool=None, input\_data=None)**

Execute the tool with the input data specified and test the results against the reference data. This effectively tests the tool execution.

**Parameters**

- **reference\_data\_dir** (*str*) – The path or vfs url of reference data to compare with
- **source\_data** (*dict*) – The source data to use

**url**

URL to website where this tool can be downloaded from

**version**

Version of the tool, not of the underlying software

**version Module**

Module containing the class that represent versions

**class fastr.core.version.Version(\*version)**

Bases: *tuple*

Class representing a software version definition. Allows for sorting and extraction of parts.

```
__dict__ = mappingproxy({'__module__': 'fastr.core.version', '__doc__': '\n Class
representing a software version definition. Allows for sorting and\n extraction of
parts.\n ', 'version_matcher':
re.compile('(\d+)\.(\d+)((?:\.\d+)+)?([_\\-\\.])?(a(?:\d)|b(?:\d)|alpha(?:\d)
|beta(?:\d)|rc(?:\d)|r(?:\d))?(\\d+)?([a-zA-Z0-9\\-_\\.]*)'),
'date_version_matcher': re.compile('(\d+)-(\d+)-(\d+)([_\\-\\.])?(.*)'),
'__new__': <staticmethod object>, '__str__': <function Version.__str__>,
'__repr__': <function Version.__repr__>, 'major': <property object>, 'minor':
<property object>, 'extra': <property object>, 'extra_string': <property object>,
'status': <property object>, 'build': <property object>, 'suffix': <property
object>, '__dict__': <attribute '__dict__' of 'Version' objects>,
'__annotations__': {}})

__module__ = 'fastr.core.version'
```

**static** `__new__`(*cls*, \**version*)

Class containing a version

Can be constructed by:

```
Version( 'major.$minor.$extra[0].$extra[1]$separator$status$build$suffix' )
Version( major, minor, extra, status, build, suffix, seperator )
Version( (major, minor, extra, status, build, suffix, seperator) )
Version( [major, minor, extra, status, build, suffix, seperator] )
```

#### Parameters

- **major** (*int*) – interger giving major version
- **minor** (*int*) – is an integer (required)
- **extra** (*list of int*) – is a list of integers
- **status** (*str*) – can be “a”, “alpha”, “b”, “beta”, “rc”, or “r”
- **build** (*int*) – is an integer
- **suffix** (*str*) – can contain any combination of alpha-numeric character and “.-”
- **seperator** (*str*) – is any of “.”, “-”, or “\_”, which is located between \$extra and \$build

---

**Note:** The method based on strings is the recommended method. For strings the major and minor version are required, where for tuple and list constructors all seven elements are optional.

---

Examples:

```
>>> a = Version('0.1')
>>> print(tuple(a))
(0, 1, None, None, None, '', None)
>>> b = Version('2.5.3-rc2')
>>> print(tuple(b))
(2, 5, [3], 'rc', 2, '', '-')
>>> c = Version('1.2.3.4.5.6.7-beta8_with_suffix')
>>> print(tuple(c))
(1, 2, [3, 4, 5, 6, 7], 'beta', 8, '_with_suffix', '-')
```

**\_\_repr\_\_**()

Return a in-editor representation of the version

**Return type**

*str*

**\_\_str\_\_**()

Return a string representation of the version

**Return type**

*str*

**property build:** *int*

the build number, this is following the status (e.g. for 3.2-beta4, this would be 4)

**Return type**

*int*



```
date_version_matcher = re.compile('(\\d+)-(\\d+)-(\\d+)([_\\-\\.])?(.*)')

property extra: Tuple[int]
    extra version extension as a list

    Return type
        Tuple[int]

property extra_string: str
    extra version extension as a string

    Return type
        str

property major: int
    major version

    Return type
        int

property minor: int
    minor version

    Return type
        int

property status: str
    the status of the version (a, alpha, b, beta, rc or r)

    Return type
        str

property suffix: str
    the remainder of the version which was not formatted in a known way

    Return type
        str

version_matcher =
re.compile('(\\d+)[_\\.](\\d+)((?:[\\-\\.\\d]+)?([_\\-\\.])?(a(?:=\\d)|b(?:=\\d)|alpha(?:=\\d)|beta(?:=\\d)|rc(?:=\\d)|r(?:=\\d))?(\\d+)?([a-zA-Z0-9[_\\-\\.])*')
```

## vfs Module

This module contains the virtual file system code. This is internally used object as used as base class for the IOPlugin.

```
class fastr.core.vfs.VirtualFileSystem
```

Bases: object

The virtual file system class. This is an IOPlugin, but also heavily used internally in fastr for working with directories. The VirtualFileSystem uses the `vfs://` url scheme.

A typical virtual filesystem url is formatted as `vfs://mountpoint/relative/dir/from/mount.ext`

Where the mountpoint is defined in the *Config file*. A list of the currently known mountpoints can be found in the `fastr.config` object

```
>>> fastr.config.mounts
{'example_data': '/home/username/fastr-feature-documentation/fastr/fastr/examples/
data',
 'home': '/home/username/',
 'tmp': '/home/username/FastrTemp'}
```

This shows that a url with the mount home such as `vfs://home/tempdir/testfile.txt` would be translated into `/home/username/tempdir/testfile.txt`.

There are a few default mount points defined by Fastr (that can be changed via the config file).

mountpoint	default location
home	the users home directory ( <code>expanduser('~')</code> )
tmp	the fastr temporary dir, defaults to <code>tempfile.gettempdir()</code>
example_data	the fastr example data directory, defaults <code>\$FASTRDIR/example/data</code>

```
__dict__ = mappingproxy({'__module__': 'fastr.core.vfs', '__doc__': "\n The
virtual file system class. This is an IOPlugin, but also heavily used\n internally
in fastr for working with directories. The VirtualFileSystem\n uses the ``vfs://``
url scheme.\n\n A typical virtual filesystem url is formatted as
``vfs://mountpoint/relative/dir/from/mount.ext``\n\n Where the ``mountpoint`` is
defined in the :ref:`config-file`. A list of\n the currently known mountpoints can
be found in the ``fastr.config`` object\n\n .. code-block:: python\n\n >>>
fastr.config.mounts\n {'example_data':
'/home/username/fastr-feature-documentation/fastr/fastr/examples/data',\n 'home':
'/home/username/',\n 'tmp': '/home/username/FastrTemp'}\n\n This shows that a url
with the mount ``home`` such as\n ``vfs://home/tempdir/testfile.txt`` would be
translated into\n ``/home/username/tempdir/testfile.txt``.\n\n There are a few
default mount points defined by Fastr (that can be changed\n via the config
file).\n\n
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
n | mountpoint | default location |\n
+=====+=====+=====+=====+=====+=====+=====+=====+=====+=====+
n | home | the users home directory (:py:func:`expanduser('~')`
<os.path.expanduser>`)|\n
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
n | tmp | the fastr temporary dir, defaults to ``tempfile.gettempdir()`` |\n
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
n | example_data | the fastr example data directory, defaults
``$FASTRDIR/example/data`` |\n
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
n\n ", '_status': (<PluginState.loaded: '\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m'>,
'), 'abstract': False, '__init__': <function VirtualFileSystem.__init__>,
'scheme': <property object>, 'setup': <function VirtualFileSystem.setup>,
'fetch_url': <function VirtualFileSystem.fetch_url>, 'fetch_value': <function
VirtualFileSystem.fetch_value>, 'put_url': <function VirtualFileSystem.put_url>,
'put_value': <function VirtualFileSystem.put_value>, 'expand_url': <function
VirtualFileSystem.expand_url>, 'expand_network_scope': <function
VirtualFileSystem.expand_network_scope>, 'url_to_path': <function
VirtualFileSystem.url_to_path>, 'path_to_url': <function
VirtualFileSystem.path_to_url>, 'copy_file_dir': <staticmethod object>,
'_correct_separators': <staticmethod object>, '__dict__': <attribute '__dict__' of
'VirtualFileSystem' objects>, '__weakref__': <attribute '__weakref__' of
'VirtualFileSystem' objects>, '__annotations__': {}})
```

**\_\_init\_\_()**

Instantiate the VFS plugin

**Returns**

the VirtualFileSystem plugin

**\_\_module\_\_** = 'fastr.core.vfs'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**abstract** = False

**static copy\_file\_dir**(*inpath*, *outpath*)

Helper function, copies a file or directory not caring what the inpath actually is

**Parameters**

- **inpath** – path of the things to be copied
- **outpath** – path of the destination

**Returns**

the result of `shutil.copy2` or `shutil.copytree` (depending on inpath pointing to a file or directory)

**expand\_network\_scope**(*value*, *network\_scope=None*)

**expand\_url**(*url*)

Try to expand the url. For vfs with will return the original url.

**Parameters**

**url** – url to expand

**Returns**

the expanded url (same as url)

**fetch\_url**(*inurl*, *outpath*)

Fetch the files from the vfs.

**Parameters**

- **inurl** – url to the item in the data store, starts with `vfs://`
- **outpath** – path where to store the fetch data locally

**fetch\_value**(*inurl*)

Fetch a value from an external vfs file.

**Parameters**

**inurl** – url of the value to read

**Returns**

the fetched value

**path\_to\_url**(*path*, *mountpoint=None*, *scheme=None*)

Construct an url from a given mount point and a relative path to the mount point.

**Parameters**

**path** (*str*) – the path to find the url for

**Mountpoint str**

mountpoint the url should be under

#### Returns

url of the

**put\_url**(*inpath*, *outurl*)

Put the files to the external data store.

#### Parameters

- **inpath** – path of the local data
- **outurl** – url to where to store the data, starts with `vfs://`

**put\_value**(*value*, *outurl*)

Put the value in the external data store.

#### Parameters

- **value** – value to store
- **outurl** – url to where to store the data, starts with `vfs://`

**property scheme**

**setup**()

The plugin setup, does nothing but needs to be implemented

**url\_to\_path**(*url*, *scheme=None*)

Get the path to a file from a vfs url

#### Parameters

**url** (*str*) – url to get the path for

#### Returns

the matching path

#### Return type

*str*

#### Raises

- ***FastrMountUnknownError*** – if the mount in url is unknown
- ***FastrUnknownURLSchemeError*** – if the url scheme is not correct

Example (the mountpoint tmp points to /tmp):

```
>>> fastr.vfs.url_to_path('vfs://tmp/file.ext')
'/tmp/file.ext'
```

## Subpackages

**test Package**

**test Package**

**test\_datatypemanager Module**

**test\_dimension Module**

**test\_samples Module**

**test\_tool Module**

**test\_version Module**

**test\_vfs Module**

**data Package**

**data Package**

Package containig data related modules

**url Module**

Module providing tools to parse and create valid urls and paths.

usage example:

When in fastr.config under the mounts section the data mount is set to /media/data, you will get the following. ..  
code-block:: python

```
>>> from fastr.data.url import get_path_from_url
>>> get_path_from_url('vfs://data/temp/blaatl.png')
'/media/data/temp/blaatl.png'
```

**fastr.data.url.basename(*url*)**

Get basename of url

**Parameters**

**url** (*str*) – the url

**Returns**

the basename of the path in the url

**fastr.data.url.create\_vfs\_url(*mountpoint*, *path*)**

Construct an url from a given mount point and a relative path to the mount point.

**Parameters**

- **mountpoint** (*str*) – the name of the mountpoint
- **path** (*str*) – relative path from the mountpoint

**Returns**

the created vfs url

**fastr.data.url.dirname(*url*)**

Get the dirname of the url

**Parameters**

**url** (*str*) – the url

**Returns**

the dirname of the path in the url

`fastr.data.url.dirurl(url)`

Get the a new url only having the dirname as the path

**Parameters**

**url** (*str*) – the url

**Returns**

the modified url with only dirname as path

`fastr.data.url.full_split(urlpath)`

Split the path in the url in a list of parts

**Parameters**

**urlpath** – the url path

**Returns**

a list of parts

`fastr.data.url.get_path_from_url(url)`

Get the path to a file from a url. Currently supports the `file://` and `vfs://` scheme's

Examples:

```
>>> url.get_path_from_url('vfs://neurodata/user/project/file.ext')
'Y:\neuro3\user\project\file.ext'

>>> 'file:///d:/data/project/file.ext'
'd:\data\project\file.ext'
```

**Warning:** `file://` will not function cross platform and is mainly for testing

`fastr.data.url.get_url_scheme(url)`

Get the schem of the url

**Parameters**

**url** (*str*) – url to extract scheme from

**Returns**

the url scheme

**Return type**

*str*

`fastr.data.url.isurl(string)`

Check if string is a valid url

**Parameters**

**string** (*str*) – potential url

**Returns**

flag indicating if string is a valid url

`fastr.data.url.join(url, *p)`

Join the path in the url with p

**Parameters**

- **url** (*str*) – the base url to join with

- **p** – additional parts of the path

**Returns**

the url with the parts added to the path

```
fastr.data.url.normurl(url)
```

Normalized the path of the url

**Parameters**

**url** (*str*) – the url

**Returns**

the normalized url

```
fastr.data.url.register_url_scheme(scheme)
```

Register a custom scheme to behave http like. This is needed to parse all things properly.

```
fastr.data.url.split(url)
```

Split a url in a url with the dirname and the basename part of the path of the url

**Parameters**

**url** (*str*) – the url

**Returns**

a tuple with (dirname\_url, basename)

**datatypes Package****datatypes Package**

The datatypes module holds all DataTypes generated by fastr and all the base classes for these datatypes.

```
class fastr.datatypes.AnyFile(value=None, format_=None)
```

Bases: [TypeGroup](#)

Special Datatype in fastr that is a TypeGroup with all known DataTypes as its members.

```
__abstractmethods__ = frozenset({})
```

```
__module__ = 'fastr.datatypes'
```

```
description: str = 'TypeGroup AnyFile\nAnyFile (AnyFile) is a group of consisting
of all URLTypes known by fastr, currently:\n - <URLType: AnalyzeImageFile class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: Directory class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: NiftiImageFileCompressed
class [\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: ProvNFile class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: NrrdImageFile class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: FilePrefix class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: MetaImageFile class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: JsonFile class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: TifImageFile class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: TxtFile class
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: NiftiImageFileUncompressed
class [\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>'
```

Description of the DataType

```
class fastr.datatypes.AnyType(value=None, format_=None)
```

Bases: [TypeGroup](#)

Special Datatype in fastr that is a TypeGroup with all known DataTypes as its members.

```
__abstractmethods__ = frozenset({})
```

```
__module__ = 'fastr.datatypes'
```

```
description: str = 'TypeGroup AnyType\nAnyType (AnyType) is a group of consisting  
of all DataTypes known by fastr, currently:\n - <ValueType: Int class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: AnalyzeImageFile class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: TxtFile class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: Directory class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <ValueType: Float class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: NiftiImageFileCompressed  
class [\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <ValueType: String class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <ValueType: UnsignedInt class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: ProvNFile class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: NrrdImageFile class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: FilePrefix class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <DataType: Missing class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <DataType: Deferred class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <ValueType: Boolean class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: JsonFile class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: TifImageFile class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: MetaImageFile class  
[\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>\n - <URLType: NiftiImageFileUncompressed  
class [\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m]>'
```

Description of the DataType

```
class fastr.datatypes.BaseDataType(value=None, format_=None)
```

Bases: [BasePlugin](#)

The base class for all datatypes in the fastr type system.

```
__abstractmethods__ = frozenset({'__init__'})
```

```
__annotations__ = {'description': <class 'str'>, 'filename': <class 'str'>,  
'version': <class 'fastr.core.version.Version'>}
```

```
__eq__(other)
```

Test the equality of two DataType objects

**Parameters**

**other** ([DataType](#)) – the object to compare against

**Returns**

flag indicating equality

**Return type**

[bool](#)

```
__getstate__()
```

```
__hash__ = None
```



**abstract \_\_init\_\_**(*value=None, format\_=None*)

The BaseDataType constructor.

**Parameters**

- **value** – value to assign to the new BaseDataType object
- **format** – the format used for the ValueType

**Returns**

new BaseDataType object

**Raises**

*FastrNotImplementedError* – if *id*, *name*, *version* or *description* is None

**\_\_module\_\_** = 'fastr.datatypes'

**\_\_ne\_\_**(*other*)

Test if two objects are not equal. This is by default done by negating the **\_\_eq\_\_** operator

**Parameters**

**other** (DataType) – the object to compare against

**Returns**

flag indicating equality

**Return type**

bool

**\_\_reduce\_ex\_\_**(\*args, \*\*kwargs)

helper for pickle

**\_\_repr\_\_**()

Returns string representation of the BaseDataType

**Returns**

string representation

**Return type**

str

**\_\_setstate\_\_**(*state*)

**\_\_str\_\_**()

Returns the string version of the BaseDataType

**Returns**

string version

**Return type**

str

**checksum**()

Generate a checksum for the value of this DataType

**Returns**

the checksum of the value

**Return type**

str

```
description: str = ''
    Description of the DataType

dot_extension = None

extension = None
    Extension related to the Type

filename: str = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/
lib/python3.6/site-packages/fastr/datatypes/__init__.py'

fullid = 'fastr://types/BaseDataType'

id = 'BaseDataType'

classmethod isinstance(value)
    Indicate whether value is an instance for this DataType.

    Returns
        the flag indicating the value is of this DataType

    Return type
        bool

name = 'BaseDataType'

parent = DataTypeManager AnalyzeImageFile : <URLType: AnalyzeImageFile> AnyFile :
<TypeGroup: AnyFile> AnyType : <TypeGroup: AnyType> Boolean : <ValueType:
Boolean> Deferred : <DataType: Deferred> Directory : <URLType: Directory>
FilePrefix : <URLType: FilePrefix> Float : <ValueType: Float> ITKImageFile :
<TypeGroup: ITKImageFile> Int : <ValueType: Int> JsonFile : <URLType: JsonFile>
MetaImageFile : <URLType: MetaImageFile> Missing : <DataType: Missing>
NiftiImageFile : <TypeGroup: NiftiImageFile> NiftiImageFileCompressed : <URLType:
NiftiImageFileCompressed> NiftiImageFileUncompressed : <URLType:
NiftiImageFileUncompressed> NrrdImageFile : <URLType: NrrdImageFile> Number :
<TypeGroup: Number> ProvnFile : <URLType: ProvnFile> String : <ValueType:
String> TifImageFile : <URLType: TifImageFile> TxtFile : <URLType: TxtFile>
UnsignedInt : <ValueType: UnsignedInt>

property parsed_value
    The parsed value of object instantiation of this DataType.

property raw_value
    The raw value of object instantiation of this DataType. For datatypes that override value (like Deferred)
    this is the way to access the _value field.

classmethod test()
    Define the test for the BasePluginManager. Make sure we are not one of the base classes

property valid
    A boolean flag that indicates weather or not the value assigned to this DataType is valid. This property is
    generally overwritten by implementation of specific DataTypes.

property value
    The value of object instantiation of this DataType.

version: Version = <Version: 1.0>
    Version of the DataType definition
```

**class** fastr.datatypes.DataType(*value=None, format\_=None*)

Bases: [BaseDataType](#), [Serializable](#)

This class is the base class for all DataTypes that can hold a value.

**\_\_abstractmethods\_\_** = frozenset({'\_\_init\_\_'})

**abstract** **\_\_init\_\_**(*value=None, format\_=None*)

The DataType constructor.

#### Parameters

- **value** – value to assign to the new DataType object
- **format** – the format used for the ValueType

#### Returns

new DataType object

**\_\_module\_\_** = 'fastr.datatypes'

**action**(*name*)

This function can be overwritten by subclasses to implement certain action that should be performed. For example, the *Directory* DataType has an action *ensure*. This method makes sure the Directory exists. A Tool can indicate an action that should be called for an Output which will be called before execution.

#### Parameters

**name** (*str*) – name of the action to execute

#### Returns

None

**classmethod** **deserialize**(*doc, \_=None*)

Classmethod that returns an object constructed based on the str/dict (or OrderedDict) representing the object

#### Parameters

**doc** (*dict*) – the state of the object to create

#### Return type

[DataType](#)

#### Returns

newly created object (of datatype indicated by the doc)

**serialize**()

Method that returns a dict structure with the datatype the object.

#### Return type

*dict*

#### Returns

serialized representation of object

**class** fastr.datatypes.DataTypeManager

Bases: [BasePluginManager](#)[[Type](#)[[BaseDataType](#)]]

The DataTypeManager hold a mapping of all DataTypes in the fast system and can create new DataTypes from files/data structures.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_args\_\_** = None

`__extra__ = None`

`__init__()`

The `DataTypeManager` constructor will create a new `DataTypeManager` and populate it with all `DataTypes` it can find in the paths set in `config.types_path`.

**Returns**

the created `DataTypeManager`

`__keytransform__(key)`

Key transformation for this mapping. The key transformation allows indexing by both the `DataType` name as well as the `DataType` it self.

**Parameters**

**key** (`fastr.datatypes.BaseDataType` or `str`) – The name of the requested datatype or the datatype itself

**Returns**

The requested datatype

`__module__ = 'fastr.datatypes'`

`__next_in_mro__`

alias of `object`

`__orig_bases__ = (fastr.abc.basepluginmanager.BasePluginManager[typing.Type[fastr.datatypes.BaseDataType]],)`

`__origin__ = None`

`__parameters__ = ()`

`__subclasshook__()`

Abstract classes can override this to customize `issubclass()`.

This is invoked early on by `abc.ABCMeta.__subclasscheck__()`. It should return `True`, `False` or `NotImplemented`. If it returns `NotImplemented`, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

`__tree_hash__ = -9223366124441915666`

`create_enumtype(type_id, options, name=None)`

Create a python class based on an XML file. This function return a completely functional python class based on the contents of a `DataType` XML file.

Such a class will be of type `EnumType`.

**Parameters**

- **type\_id** (`str`) – the id of the new class
- **options** (`iterable`) – an iterable of options, each option should be `str`

**Return type**

`Type[EnumType]`

**Returns**

the newly created subclass of `EnumType`

**Raises**

`FastrTypeError` – if the options is not an iterable of `str`

**property fullid**

The fullid of the datatype manager

**get\_type(name)**

Read a type given a typename. This will scan all directories in types\_path and attempt to load the newest version of the DataType.

**Parameters**

**name** (*str*) – Name of the datatype that should be imported in the system

**Return type**

*Type*[*BaseDataType*]

**Returns**

the datatype with the requested name, or None if datatype is not found

---

**Note:** If type is already in TypeManager it will not load anything and return the already loaded version.

---

**guess\_type(value, exists=True, options=None, preferred=None)**

Guess the DataType based on a value str.

**Parameters**

- **value** (*str*) – the value to guess the type for
- **options** (*TypeGroup*, *DataType* or *tuple of DataTypes*) – The options that are allowed to be guessed from
- **exists** (*bool*) – Indicate the value exists (if file) and can be checked for validity, if false skip validity check
- **preferred** (*iterable*) – An iterable of preferred types in case multiple types match.

**Return type**

*Optional*[*Type*[*BaseDataType*]]

**Returns**

The resulting DataType or None if no match was found

**Raises**

*FastrTypeError* – if the options argument is of the wrong type

The function will first create a list of all candidate DataTypes. Subsequently, it will check for each candidate if the value would valid. If there are multiple matches, the config value for preferred types is consulted to break the ties. If non of the DataTypes are in the preferred types list, a somewhat random DataType will be picked as the most optimal result.

**has\_type(name)**

Check if the datatype with requested name exists

**Parameters**

**name** (*str*) – the name of the requested datatype

**Returns**

flag indicating if the datatype exists

**Return type**

*bool*

**static isdatatype(*item*)**

Check if item is a valid datatype for the fastr system.

**Parameters**

**item** – item to check

**Returns**

flag indicating if the item is a fastr datatype

**Return type**

bool

**match\_types(\*args, \*\*kwargs)**

Find the match between a list of DataTypes/TypeGroups, see [Resolving Datatypes](#) for details

**Parameters**

- **args** – A list of DataType/TypeGroup objects to match
- **kwargs** – A ‘preferred’ keyword argument can be used to indicate a list of DataTypes to prefer in case of ties (first has precedence over later in list)

**Returns**

The best DataType match, or None if no match is possible.

**Raises**

**FastrTypeError** – if not all args are subclasses of BaseDataType

**match\_types\_any(\*args)**

Find the match between a list of DataTypes/TypeGroups, see [Resolving Datatypes](#) for details

**Parameters**

**args** – A list of DataType/TypeGroup objects to match

**Returns**

A set with all DataTypes that match.

**Return type**

set

**Raises**

**FastrTypeError** – if not all args are subclasses of BaseDataType

**property plugin\_class**

The PluginClass of the items of the BasePluginManager

**poll\_datatype(filename)**

Poll an xml file to see if there is a definition of a datatype in it.

**Parameters**

**filename** (*str*) – path of the file to poll

**Returns**

tuple with (id, version, basetype) if a datatype is found or (None, None, None) if no datatype is found

**populate()**

Populate Manager. After scanning for DataTypes, create the AnyType and set the preferred types

**property preferred\_types**

**class** fastr.datatypes.Deferred(*value=None, format\_=None*)

Bases: *DataType*

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_getstate\_\_**()

**\_\_init\_\_**(*value=None, format\_=None*)

The Deferred constructor.

#### Parameters

- **value** – value to assign to the new *DataType* object
- **format** – This is ignore but here for compatibility

#### Returns

new Deferred object

**\_\_module\_\_** = 'fastr.datatypes'

**\_\_repr\_\_**()

Returns string representation of the *BaseDataType*

#### Returns

string represenation

#### Return type

*str*

**\_\_setstate\_\_**(*state*)

**checksum**()

Generate a checksum for the value of this *DataType*

#### Returns

the checksum of the value

#### Return type

*str*

**property** job

**classmethod** lookup(*value*)

Look up the deferred target and return that object

#### Param

value

#### Returns

The value the deferred points to

#### Return type

*DataType*

#### Raises

- *FastrKeyError* – if the deferred is not available (yet)
- *FastrValueError* – if the value is not a valid deferred url

**property** parsed\_value

The value of object instantiation of this *DataType*.

**property provenance**

**property target**

Target object for this deferred.

**Raises**

- *FastrKeyError* – if the deferred is not available (yet)
- *FastrValueError* – if the value is not a valid deferred url

**property value**

The value of object instantiation of this DataType.

**class** `fastr.datatypes.EnumType(value=None,format_=None)`

Bases: *DataType*

The EnumType is the base for DataTypes that can have a value which is an option from a predefined set of possibilities (similar to an enum type in many programming languages).

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_init\_\_**(value=None,format\_=None)

The EnumType constructor.

**Parameters**

- **value** – value to assign to the new EnumType object
- **format** – the format used for the ValueType

**Returns**

new EnumType object

**Raises**

*FastrDataTypeNotInstantiableError* – if not subclassed

**\_\_module\_\_** = `'fastr.datatypes'`

**\_\_reduce\_ex\_\_**(\*args,\*\*kwargs)

helper for pickle

**description:** `str = 'EnumType (EnumType) is a enumerate type with options:\n\nEnumType can take the value of any of the option, but any other value is considered invalid.'`

Description of the DataType

**options** = `frozenset({})`

**version:** `Version = <Version: 1.0>`

Enums always have version 1.0

**class** `fastr.datatypes.Missing(*args,**kwargs)`

Bases: *DataType*

Singleton DataType to annotate missing data

**\_\_abstractmethods\_\_** = `frozenset({})`



```
__init__(_=None, __=None)
```

The DataType constructor.

#### Parameters

- **value** – value to assign to the new DataType object
- **format** – the format used for the ValueType

#### Returns

new DataType object

```
__module__ = 'fastr.datatypes'
```

```
static __new__(cls, *args, **kwargs)
```

```
value = 'MISSING'
```

```
class fastr.datatypes.TypeGroup(value=None, format_=None)
```

Bases: [BaseDataType](#)

The TypeGroup is a special DataType that does not hold a value of its own but is used to group a number of DataTypes. For example ITK has a list of supported file formats that all tools build on ITK support. A group can be used to conveniently specify this in multiple Tools that use the same set DataTypes.

```
__abstractmethods__ = frozenset({'_members'})
```

```
__init__(value=None)
```

Dummy constructor. TypeGroups are not instantiable and cannot hold a value of its own.

#### Raises

[FastrDataTypeNotInstantiableError](#) – if called

```
__module__ = 'fastr.datatypes'
```

```
static __new__(cls, value=None, format_=None)
```

Instantiate a TypeGroup. This will for match the value to the best matching type and instantiate that. Not that the returned object will not be of type TypeGroup but one of the TypeGroup members.

```
classmethod isinstance(value)
```

Indicate whether value is an instance for this DataType.

#### Returns

the flag indicating the value is of this DataType

#### Return type

[bool](#)

#### members

A descriptor that can act like a property for a class.

#### preference

A descriptor that can act like a property for a class.

```
class fastr.datatypes.URLType(value=None, format_=None)
```

Bases: [DataType](#)

The URLType is the base for DataTypes that point to a resource somewhere else (typically a filesystem). The true value is actually the resource referenced by the value in this object.

```
__abstractmethods__ = frozenset({})
```

`__eq__(other)`

Test the equality of two `DataType` objects

**Parameters**

**other** (`URLType`) – the object to compare against

**Returns**

flag indicating equality

**Return type**

`bool`

`__hash__ = None`

`__init__(value=None, format_=None)`

The `URLType` constructor

**Parameters**

- **value** – value to assign to the new `URLType`
- **format** – the format used for the `ValueType`

**Returns**

new `URLType` object

`__module__ = 'fastr.datatypes'`

`checksum()`

Return the checksum of this `URL` type

**Returns**

checksum string

**Return type**

`str`

**classmethod** `content(ival, outval=None)`

Give the contents of a `URLType`, this is generally useful for filetypes that consists of multiple files (e.g. `AnalyzeImageFile`, `DICOM`). The value will indicate the main file, and the contents function can determine all files that form a single data value.

**Parameters**

- **ival** – a value to figure out contents for this type
- **outval** – the place where the copy should point to

**Returns**

a list of all files part of the value (e.g. header and data file)

**Return type**

`list`

**property** `parsed_value`

The parsed value of object instantiation of this `DataType`.

**property** `valid`

A boolean flag that indicates weather or not the value assigned to this `DataType` is valid. This property is generally overwritten by implementation of specific `DataTypes`.

```
class fastr.datatypes.ValueType(value=None,format_=None)
```

Bases: [DataType](#)

The ValueType is the base for DataTypes that hold simple values (not an EnumType and not a file/URL). The values is generally represented by a string.

```
__abstractmethods__ = frozenset({})
```

```
__init__(value=None,format_=None)
```

The ValueType constructor

#### Parameters

- **value** – value to assign to the new ValueType
- **format** – the format used for the ValueType

#### Returns

new ValueType object

```
__module__ = 'fastr.datatypes'
```

```
fastr.datatypes.fastr_isinstance(obj, datatype)
```

Check if an object is of a specific datatype.

#### Parameters

- **obj** – Object to inspect
- **datatype** ([tuple](#), [BaseDataType](#)) – The datatype(s) to check

#### Returns

flag indicating object is of datatype

#### Return type

[bool](#)

## execution Package

### execution Package

This package contains all modules related directly to the execution

### basenoderun Module

```
class fastr.execution.basenoderun.BaseNodeRun
```

Bases: [Updateable](#), [Serializable](#)

```
NODE_RUN_MAP = {'AdvancedFlowNode': <class
'fastr.execution.flownoderun.AdvancedFlowNodeRun'>, 'ConstantNode': <class
'fastr.execution.sourcenoderun.ConstantNodeRun'>, 'FlowNode': <class
'fastr.execution.flownoderun.FlowNodeRun'>, 'MacroNode': <class
'fastr.execution.macronoderun.MacroNodeRun'>, 'Node': <class
'fastr.execution.noderun.NodeRun'>, 'SinkNode': <class
'fastr.execution.sinknoderun.SinkNodeRun'>, 'SourceNode': <class
'fastr.execution.sourcenoderun.SourceNodeRun'>}
```

```
NODE_RUN_TYPES = {'AdvancedFlowNodeRun': <class
'fastr.execution.flownoderun.AdvancedFlowNodeRun'>, 'ConstantNodeRun': <class
'fastr.execution.sourcenoderun.ConstantNodeRun'>, 'FlowNodeRun': <class
'fastr.execution.flownoderun.FlowNodeRun'>, 'MacroNodeRun': <class
'fastr.execution.macronoderun.MacroNodeRun'>, 'NodeRun': <class
'fastr.execution.noderun.NodeRun'>, 'SinkNodeRun': <class
'fastr.execution.sinknoderun.SinkNodeRun'>, 'SourceNodeRun': <class
'fastr.execution.sourcenoderun.SourceNodeRun'>}

__abstractmethods__ = frozenset({'_update'})

classmethod __init_subclass__(**kwargs)
    Register nodes in class for easily location

__module__ = 'fastr.execution.basenoderun'
```

## environmentmodules Module

This module contains a class to interact with EnvironmentModules

```
class fastr.execution.environmentmodules.EnvironmentModules(protected=None)
```

Bases: `object`

This class can control the module environments in python. It can list, load and unload environmentmodules. These modules are then used if subprocess is called from python.

```
__dict__ = mappingproxy({'__module__': 'fastr.execution.environmentmodules',
'__doc__': '\n This class can control the module environments in python. It can
list, load\n and unload environmentmodules. These modules are then used if
subprocess is\n called from python.\n ', '_module_settings_loaded': False,
'_module_settings_warning': 'Cannot find Environment Modules home directory
(environment variables not setup properly?)', '__init__': <function
EnvironmentModules.__init__>, '__repr__': <function EnvironmentModules.__repr__>,
'sync': <function EnvironmentModules.sync>, '_sync_loaded': <function
EnvironmentModules._sync_loaded>, '_sync_avail': <function
EnvironmentModules._sync_avail>, '_module': <function EnvironmentModules._module>,
'totuple_modvalue': <staticmethod object>, 'tostring_modvalue': <staticmethod
object>, '_run_commands_string': <function
EnvironmentModules._run_commands_string>, 'loaded_modules': <property object>,
'avail_modules': <property object>, 'avail': <function EnvironmentModules.avail>,
'isloaded': <function EnvironmentModules.isloaded>, 'load': <function
EnvironmentModules.load>, 'unload': <function EnvironmentModules.unload>, 'reload':
<function EnvironmentModules.reload>, 'swap': <function EnvironmentModules.swap>,
'clear': <function EnvironmentModules.clear>, '__dict__': <attribute '__dict__' of
'EnvironmentModules' objects>, '__weakref__': <attribute '__weakref__' of
'EnvironmentModules' objects>, '__annotations__': {}})
```

```
__init__(protected=None)
```

Create the environmentmodules control object

### Parameters

**protected** (*list*) – list of modules that should never be unloaded

### Returns

newly created EnvironmentModules

**\_\_module\_\_** = 'fastr.execution.environmentmodules'

**\_\_repr\_\_**()

Return repr(self).

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**avail**(*namestart=None*)

Print available modules in same way as commandline version

**Parameters**

**namestart** – filter on modules that start with namestart

**property avail\_modules**

List of available modules

**clear**()

Unload all modules (except the protected modules as they cannot be unloaded). This should result in a clean environment.

**isloaded**(*module*)

Check if a specific module is loaded

**Parameters**

**module** – module to check

**Returns**

flag indicating the module is loaded

**load**(*module*)

Load specified module

**Parameters**

**module** – module to load

**property loaded\_modules**

List of currently loaded modules

**reload**(*module*)

Reload specified module

**Parameters**

**module** – module to reload

**swap**(*module1, module2*)

Swap one module for another one

**Parameters**

- **module1** – module to unload
- **module2** – module to load

**sync**()

Sync the object with the underlying environment. Re-checks the available and loaded modules

**static tostring\_modvalue**(*value*)

Turn a representation of a module into a string representation

**Parameters**

**value** – module representation (either str or tuple)

**Returns**

string representation

**static totuple\_modvalue**(*value*)

Turn a representation of a module into a tuple representation

**Parameters**

**value** – module representation (either str or tuple)

**Returns**

tuple representation (name, version, default)

**unload**(*module*)

Unload specified module

**Parameters**

**module** – module to unload

**class** fastr.execution.environmentmodules.**ModuleSystem**(*value*)

Bases: [Enum](#)

An enumeration.

**\_\_module\_\_** = 'fastr.execution.environmentmodules'

**envmod** = 'enviromentmodules'

**lmod** = 'Lmod'

## executionscript Module

The executionscript is the script that wraps around a tool executable. It takes a job, builds the command, executes the command (while profiling it) and collects the results.

**fastr.execution.executionscript.execute\_job**(*job*)

Execute a Job and save the result to disk

**Parameters**

**job** – the job to execute

**fastr.execution.executionscript.main**(*joblist=None*)

This is the main code. Wrapped inside a function to avoid the variables being seen as globals and to shut up pylint. Also if the joblist argument is given it can run any given job, otherwise it takes the first command line argument.

## flownoderun Module

**class** fastr.execution.flownoderun.**AdvancedFlowNodeRun**(*node, parent*)

Bases: [FlowNodeRun](#)

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_module\_\_** = 'fastr.execution.flownoderun'

**set\_result**(*job*, *failed\_annotation*)

Incorporate result of a job into the FlowNodeRun.

**Parameters**

**job** (*Type*) – job of which the result to store

**class** fastr.execution.flownoderun.**FlowNodeRun**(*node*, *parent*)

Bases: [NodeRun](#)

A Flow NodeRun is a special subclass of Nodes in which the amount of samples can vary per Output. This allows non-default data flows.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_module\_\_** = 'fastr.execution.flownoderun'

**property blocking**

A FlowNodeRun is (for the moment) always considered blocking.

**Returns**

True

**property dimnames**

Names of the dimensions in the NodeRun output. These will be reflected in the SampleIdList of this NodeRun.

**property outputsizes**

Size of the outputs in this NodeRun

**set\_result**(*job*, *failed\_annotation*)

Incorporate result of a job into the FlowNodeRun.

**Parameters**

**job** (*Type*) – job of which the result to store

## inputoutputrun Module

Classes for arranging the input and output for nodes.

Exported classes:

Input – An input for a node (holding datatype). Output – The output of a node (holding datatype and value). ConstantOutput – The output of a node (holding datatype and value).

**Warning:** Don't mess with the Link, Input and Output internals from other places. There will be a huge chances of breaking the network functionality!

**class** fastr.execution.inputoutputrun.**AdvancedFlowOutputRun**(*node\_run*, *template*)

Bases: [OutputRun](#)

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_module\_\_** = 'fastr.execution.inputoutputrun'

```
class fastr.execution.inputoutputrun.BaseInputRun(node_run, template)
```

Bases: [HasSamples](#), [BaseInput](#)

Base class for all inputs runs.

```
__abstractmethods__ = frozenset({'__getitem__', '_update', 'dimensions', 'fullid',
'itersubinputs'})
```

```
__init__(node_run, template)
```

Instantiate a BaseInput

#### Parameters

- **node** – the parent node the input/output belongs to.
- **description** – the ParameterDescription describing the input/output.

#### Returns

the created BaseInput

#### Raises

- [FastrTypeError](#) – if description is not of class ParameterDescription
- [FastrDataTypeNotAvailableError](#) – if the DataType requested cannot be found in the `fastr.types`

```
__module__ = 'fastr.execution.inputoutputrun'
```

```
abstract itersubinputs()
```

Iterator over the SubInputs

#### Returns

iterator

example:

```
>>> for subinput in input_a.itersubinputs():
    print subinput
```

```
class fastr.execution.inputoutputrun.InputRun(node_run, template)
```

Bases: [BaseInputRun](#)

Class representing an input of a node. Such an input will be connected to the output of another node or the output of an constant node to provide the input value.

```
__abstractmethods__ = frozenset({})
```

```
__getitem__(key)
```

Retrieve an item from this Input.

#### Parameters

**key** (str, SampleId or tuple) – the key of the requested item, can be a key str, sample index tuple or a SampleId

#### Returns

the return value depends on the requested key. If the key was an int the corresponding [SubInput](#) will be returned. If the key was a SampleId or sample index tuple, the corresponding SampleItem will be returned.

#### Return type

SampleItem or [SubInput](#)



**Raises**

- *FastrTypeError* – if key is not of a valid type
- *FastrKeyError* – if the key is not found

**\_\_getstate\_\_()**

Retrieve the state of the Input

**Returns**

the state of the object

**Rtype dict****\_\_init\_\_(node\_run, template)**

Instantiate an input.

**Parameters**

**template** – the Input that the InputRun is based on

**\_\_module\_\_ = 'fastr.execution.inputoutputrun'****\_\_setstate\_\_(state)**

Set the state of the Input by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**\_\_str\_\_()**

Get a string version for the Input

**Returns**

the string version

**Return type**

*str*

**cardinality(key=None, job\_data=None)**

Cardinality for an Input is the sum the cardinalities of the SubInputs, unless defined otherwise.

**Parameters**

**key** (tuple of int or *SampleId*) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

*int*, *sympy.Symbol*, or *None*

**property datatype**

The datatype of this Input

**property dimensions**

The size of the sample collections that can accessed via this Input.

**property fullid**

The full defining ID for the Input

### **get\_sourced\_nodes()**

Get a list of all *Nodes* connected as sources to this Input

#### **Returns**

list of all connected *Nodes*

#### **Return type**

list

### **get\_sourced\_outputs()**

Get a list of all *Outputs* connected as sources to this Input

#### **Returns**

tuple of all connected *Outputs*

#### **Return type**

tuple

### **get\_subinput\_cardinality(index, key=None, job\_data=None)**

Cardinality for a SubInput

#### **Parameters**

- **index** (*int*) – index for a specific sample
- **key** (tuple of int or *SampleId*) – key for a specific sample, can be sample index or id

#### **Returns**

the cardinality

#### **Return type**

*int*, *sympy.Symbol*, or *None*

### **index(value)**

Find index of a SubInput

#### **Parameters**

**value** (*SubInput*) – the *SubInput* to find the index of

#### **Returns**

key

#### **Return type**

*int*, *str*

### **property input\_group**

The id of the InputGroup this Input belongs to.

### **insert(index)**

Insert a new SubInput at index in the sources list

#### **Parameters**

**key** (*int*) – positive integer for position in *\_source* list to insert to

#### **Returns**

newly inserted *SubInput*

#### **Return type**

*SubInput*

### **itersubinputs()**

Iterate over the *SubInputs* in this Input.

**Returns**

iterator yielding *SubInput*

example:

```
>>> for subinput in input_a.itorsubinputs():
      print subinput
```

**remove(value)**

Remove a SubInput from the SubInputs list.

**Parameters**

**value** (*SubInput*) – the *SubInput* to removed from this Input

**property source**

The mapping of *SubInputs* that are connected and have more than 0 elements.

**class** fastr.execution.inputoutputrun.**MacroOutputRun**(*node\_run, template*)

Bases: *OutputRun*

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_module\_\_** = 'fastr.execution.inputoutputrun'

**property dimensions**

The dimensions has to be implemented by any subclass. It has to provide a tuple of Dimensions.

**Returns**

dimensions

**Return type**

tuple

**class** fastr.execution.inputoutputrun.**NamedSubinputRun**(*parent*)

Bases: *InputRun*

A named subinput for cases where the value of an input is mapping.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_getitem\_\_**(*key*)

Retrieve an item (a SubInput) from this NamedSubInput.

**Parameters**

**key** (int) – the key of the requested item

**Return type**

Union[*SubInputRun*, *SampleItem*]

**Returns**

The *SubInput* corresponding with the key will be returned.

**Raises**

- *FastrTypeError* – if key is not of a valid type
- *FastrKeyError* – if the key is not found

**\_\_init\_\_**(*parent*)

Instantiate an input.

**Parameters**

**template** – the Input that the InputRun is based on

**\_\_module\_\_** = 'fastr.execution.inputoutputrun'

**\_\_str\_\_**()

Get a string version for the NamedSubInput

**Returns**

the string version

**Return type**

str

**property fullid**

The full defining ID for the NamedSubInputRun

**property item\_index**

**class** fastr.execution.inputoutputrun.**OutputRun**(node\_run, template)

Bases: [BaseOutput](#), [ContainsSamples](#)

Class representing an output of a node. It holds the output values of the tool ran. Output fields can be connected to inputs of other nodes.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_getitem\_\_**(key)

Retrieve an item from this Output. The returned value depends on what type of key used:

- Retrieving data using index tuple: [index\_tuple]
- Retrieving data sample\_id str: [SampleId]
- Retrieving a list of data using SampleId list: [sample\_id1, ..., sample\_idN]
- Retrieving a [SubOutput](#) using an int or slice: [n] or [n:m]

**Parameters**

**key** (int, slice, SampleId or tuple) – the key of the requested item, can be a number, slice, sample index tuple or a SampleId

**Returns**

the return value depends on the requested key. If the key was an int or slice the corresponding [SubOutput](#) will be returned (and created if needed). If the key was a SampleId or sample index tuple, the corresponding SampleItem will be returned. If the key was a list of SampleId a tuple of SampleItem will be returned.

**Return type**

[SubInput](#) or SampleItem or list of SampleItem

**Raises**

- [FastrTypeError](#) – if key is not of a valid type
- [FastrKeyError](#) – if the parent Node has not been executed

**\_\_getstate\_\_**()

Retrieve the state of the Output

**Returns**

the state of the object

**Rtype dict****\_\_init\_\_**(*node\_run*, *template*)

Instantiate an Output

**Parameters**

- **node** – the parent node the output belongs to.
- **description** – the `ParameterDescription` describing the output.

**Returns**

created Output

**Raises**

- `FastrTypeError` – if description is not of class `ParameterDescription`
- `FastrDataTypeNotAvailableError` – if the `DataType` requested cannot be found in the `fastr.types`

**\_\_module\_\_** = 'fastr.execution.inputoutputrun'**\_\_setitem\_\_**(*key*, *value*)

Store an item in the Output

**Parameters**

- **key** (tuple of int or `SampleId`) – key of the value to store
- **value** – the value to store

**Returns**

None

**Raises**`FastrTypeError` – if key is not of correct type**\_\_setstate\_\_**(*state*)

Set the state of the Output by the given state.

**Parameters****state** (*dict*) – The state to populate the object with**Returns**

None

**\_\_str\_\_**()

Get a string version for the Output

**Returns**

the string version

**Return type**`str`**property automatic**

Flag indicating that the Output is generated automatically without being specified on the command line

**cardinality**(*key=None*, *job\_data=None*)

Cardinality of this Output, may depend on the inputs of the parent Node.

**Parameters****key** (tuple of int or `SampleId`) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

`int`, `sympy.Symbol`, or `None`

**Raises**

- ***FastrCardinalityError*** – if cardinality references an invalid *Input*
- ***FastrTypeError*** – if the referenced cardinality values type cannot be case to int
- ***FastrValueError*** – if the referenced cardinality value cannot be case to int

**property datatype**

The datatype of this Output

**property fullid**

The full defining ID for the Output

**iterconvergingindices(*collapse\_dims*)**

Iterate over all data, but collapse certain dimension to create lists of data.

**Parameters**

**collapse\_dims** (*iterable of int*) – dimension to collapse

**Returns**

iterator `SampleIndex` (possibly containing slices)

**property listeners**

The list of *Links* connected to this Output.

**property preferred\_types**

The list of preferred `DataTypes` for this Output.

**property resulting\_datatype**

The `DataType` that will the results of this Output will have.

**property samples**

The `SampleCollection` of the samples in this Output. `None` if the `NodeRun` has not yet been executed. Otherwise a `SampleCollection`.

**property valid**

Check if the output is valid, i.e. has a valid cardinality

**class** `fastr.execution.inputoutputrun.SourceOutputRun(node_run, template)`

Bases: *OutputRun*

Output for a `SourceNodeRun`, this type of Output determines the cardinality in a different way than a normal `NodeRun`.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_getitem\_\_**(*item*)

Retrieve an item from this Output. The returned value depends on what type of key used:

- Retrieving data using index tuple: `[index_tuple]`
- Retrieving data `sample_id` str: `[SampleId]`
- Retrieving a list of data using `SampleId` list: `[sample_id1, ..., sample_idN]`
- Retrieving a *SubOutput* using an int or slice: `[n]` or `[n:m]`

**Parameters**

**key** (int, slice, SampleId or tuple) – the key of the requested item, can be a number, slice, sample index tuple or a SampleId

**Returns**

the return value depends on the requested key. If the key was an int or slice the corresponding *SubOutput* will be returned (and created if needed). If the key was a SampleId or sample index tuple, the corresponding SampleItem will be returned. If the key was a list of SampleId a tuple of SampleItem will be returned.

**Return type**

*SubInput* or SampleItem or list of SampleItem

**Raises**

- *FastrTypeError* – if key is not of a valid type
- *FastrKeyError* – if the parent NodeRun has not been executed

`__init__(node_run, template)`

Instantiate a FlowOutput

**Parameters**

- **node** – the parent node the output belongs to.
- **description** – the ParameterDescription describing the output.

**Returns**

created FlowOutput

**Raises**

- *FastrTypeError* – if description is not of class ParameterDescription
- *FastrDataTypeNotAvailableError* – if the DataType requested cannot be found in the `fastr.types`

`__module__ = 'fastr.execution.inputoutputrun'`

`__setitem__(key, value)`

Store an item in the Output

**Parameters**

- **key** (tuple of int or SampleId) – key of the value to store
- **value** – the value to store

**Returns**

None

**Raises**

*FastrTypeError* – if key is not of correct type

`cardinality(key=None, job_data=None)`

Cardinality of this SourceOutput, may depend on the inputs of the parent NodeRun.

**Parameters**

**key** (tuple of int or SampleId) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

`int`, `sympy.Symbol`, or `None`

**property dimensions**

The dimensions of this `SourceOutputRun`

**property linearized**

A linearized version of the sample data, this is lazily cached linearized version of the underlying `SampleCollection`.

**property ndims**

The number of dimensions in this `SourceOutput`

**property size**

The sample size of the `SourceOutput`

**class** `fastr.execution.inputoutputrun.SubInputRun(input_)`

Bases: `BaseInputRun`

This class is used by `Input` to allow for multiple links to an `Input`. The `SubInput` class can hold only a single Link to a (Sub)Output, but behaves very similar to an `Input` otherwise.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_getitem\_\_**(*key*)

Retrieve an item from this `SubInput`.

**Parameters**

**key** (`int`, `SampleId` or `SampleIndex`) – the key of the requested item, can be a number, sample index tuple or a `SampleId`

**Returns**

the return value depends on the requested key. If the key was an `int` the corresponding `SubInput` will be returned. If the key was a `SampleId` or sample index tuple, the corresponding `SampleItem` will be returned.

**Return type**

`SampleItem` or `SubInput`

**Raises**

`FastrTypeError` – if key is not of a valid type

---

**Note:** As a `SubInput` has only one `SubInput`, only requesting `int` key 0 or -1 is allowed, and it will return self

---

**\_\_getstate\_\_**()

Retrieve the state of the `SubInput`

**Returns**

the state of the object

**Rtype dict**

**\_\_init\_\_**(*input\_*)

Instantiate an `SubInput`.

**Parameters**

**input** (`Input`) – the parent of this `SubInput`.



**Returns**

the created SubInput

**\_\_module\_\_** = 'fastr.execution.inputoutputrun'

**\_\_setstate\_\_**(*state*)

Set the state of the SubInput by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**\_\_str\_\_**()

Get a string version for the SubInput

**Returns**

the string version

**Return type**

*str*

**cardinality**(*key=None, job\_data=None*)

Get the cardinality for this SubInput. The cardinality for a SubInputs is defined by the incoming link.

**Parameters**

**key** (SampleIndex or SampleId) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

*int*, *sympy.Symbol*, or *None*

**property description**

The description object of this input/output

**property dimensions**

The sample size of the SubInput

**property fullid**

The full defining ID for the SubInput

**get\_sourced\_nodes**()

Get a list of all *Nodes* connected as sources to this SubInput

**Returns**

list of all connected *Nodes*

**Return type**

*list*

**get\_sourced\_outputs**()

Get a list of all *Outputs* connected as sources to this SubInput

**Returns**

list of all connected *Outputs*

**Return type**

*list*

**property input\_group**

The id of the InputGroup this SubInputs parent belongs to.

**property item\_index****iteritems()**

Iterate over the SampleItems that are in the SubInput.

**Returns**

iterator yielding SampleItem objects

**itersubinputs()**

Iterate over SubInputs (for a SubInput it will yield self and stop iterating after that)

**Returns**

iterator yielding *SubInput*

example:

```
>>> for subinput in input_a.itersubinputs():  
    print subinput
```

**property node**

The Node to which this SubInputs parent belongs

**property source**

A list with the source *Link*. The list is to be compatible with *Input*

**property source\_output**

The *Output* linked to this SubInput

**class** fastr.execution.inputoutputrun.**SubOutputRun**(*output, index*)

Bases: *OutputRun*

The SubOutput is an Output that represents a slice of another Output.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_getitem\_\_**(*key*)

Retrieve an item from this SubOutput. The returned value depends on what type of key used:

- Retrieving data using index tuple: [index\_tuple]
- Retrieving data sample\_id str: [SampleId]
- Retrieving a list of data using SampleId list: [sample\_id1, ..., sample\_idN]
- Retrieving a *SubOutput* using an int or slice: [n] or [n:m]

**Parameters**

**key** (int, slice, SampleId or tuple) – the key of the requested item, can be a number, slice, sample index tuple or a SampleId

**Returns**

the return value depends on the requested key. If the key was an int or slice the corresponding *SubOutput* will be returned (and created if needed). If the key was a SampleId or sample index tuple, the corresponding SampleItem will be returned. If the key was a list of SampleId a tuple of SampleItem will be returned.

**Return type**

*SubInput* or SampleItem or list of SampleItem

**Raises**

*FastrTypeError* – if key is not of a valid type

**\_\_getstate\_\_()**

Retrieve the state of the SubOutput

**Returns**

the state of the object

**Rtype dict****\_\_init\_\_(output, index)**

Instantiate a SubOutput

**Parameters**

- **output** – the parent output the suboutput slices.
- **index** (*int* or *slice*) – the way to slice the parent output

**Returns**

created SubOutput

**Raises**

- *FastrTypeError* – if the output argument is not an instance of *Output*
- *FastrTypeError* – if the index argument is not an *int* or *slice*

**\_\_len\_\_()**

Return the length of the Output.

---

**Note:** In a SubOutput this is always 1.

---

**\_\_module\_\_ = 'fastr.execution.inputoutputrun'****\_\_setitem\_\_(key, value)**

A function blocking the assignment operator. Values cannot be assigned to a SubOutput.

**Raises**

*FastrNotImplementedError* – if called

**\_\_setstate\_\_(state)**

Set the state of the SubOutput by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**\_\_str\_\_()**

Get a string version for the SubOutput

**Returns**

the string version

**Return type**

*str*

**cardinality**(*key=None, job\_data=None*)

Cardinality of this SubOutput depends on the parent Output and `self.index`

**Parameters**

**key** (tuple of int or `SampleId`) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

int, `sympy.Symbol`, or `None`

**Raises**

- ***FastrCardinalityError*** – if cardinality references an invalid *Input*
- ***FastrTypeError*** – if the referenced cardinality values type cannot be case to int
- ***FastrValueError*** – if the referenced cardinality value cannot be case to int

**property datatype**

The datatype of this SubOutput

**property fullid**

The full defining ID for the SubOutput

**property indexrep**

Simple representation of the index.

**property listeners**

The list of *Links* connected to this Output.

**property node**

The `NodeRun` to which this SubOutput belongs

**property preferred\_types**

The list of preferred `DataTypes` for this SubOutput.

**property resulting\_datatype**

The `DataType` that will the results of this SubOutput will have.

**property samples**

The `SampleCollection` for this SubOutput

## job Module

This module contains the `Job` class and some related classes.

**class** `fastr.execution.job.InlineJob(*args, **kwargs)`

Bases: *Job*

Job that does not actually need to run but is used for consistency in data processing and logging.

**\_\_init\_\_**(*\*args, \*\*kwargs*)

Create a job

**Parameters**

- **node** (`fastr.planning.node.Node`) – the node the job is based on
- **sample\_id** – the id of the sample

- **sample\_index** – the index of the sample
- **input\_arguments** – the argument list
- **output\_arguments** – the argument list
- **hold\_jobs** – the jobs on which this jobs depend
- **preferred\_types** – The list of preferred types to use

**Returns**

**\_\_module\_\_** = 'fastr.execution.job'

**collect\_provenance()**

Collect the provenance for this job

**get\_result()**

Get the result of the job if it is available. Load the output file if found and check if the job matches the current object. If so, load and return the result.

**Returns**

Job after execution or None if not available

**Return type**

*Job* | None

```
class fastr.execution.job.Job(node, sample_id, sample_index, input_arguments, output_arguments,  
                             hold_jobs=None, preferred_types=None)
```

Bases: Serializable

Class describing a job.

Arguments: tool\_name - the name of the tool (str) tool\_version - the version of the tool (Version) argument - the arguments used when calling the tool (list) tmpdir - temporary directory to use to store output data hold\_jobs - list of jobs that need to finished before this job can run (list)

**COMMAND\_DUMP** = '\_\_fastr\_command\_\_.yaml'

**INFO\_DUMP** = '\_\_fastr\_extra\_job\_info\_\_.yaml'

**PROV\_DUMP** = '\_\_fastr\_prov\_\_.json'

**RESULT\_DUMP** = '\_\_fastr\_result\_\_.yaml'

**STDERR\_DUMP** = '\_\_fastr\_stderr\_\_.txt'

**STDOUT\_DUMP** = '\_\_fastr\_stdout\_\_.txt'

**\_\_getstate\_\_()**

Get the state of the job

**Returns**

job state

**Return type**

dict

```
__init__(node, sample_id, sample_index, input_arguments, output_arguments, hold_jobs=None,  
         preferred_types=None)
```

Create a job

**Parameters**

- **node** (`fastr.planning.node.Node`) – the node the job is based on
- **sample\_id** (`SampleId`) – the id of the sample
- **sample\_index** (`SampleIndex`) – the index of the sample
- **input\_arguments** (`Dict[str, SampleItem]`) – the argument list
- **output\_arguments** (`Dict[str, Dict]`) – the argument list
- **hold\_jobs** (`Optional[List[str]]`) – the jobs on which this jobs depend
- **preferred\_types** (`Optional[List]`) – The list of preferred types to use

**Returns**

**\_\_module\_\_** = 'fastr.execution.job'

**\_\_repr\_\_**()

String representation of the Job

**\_\_setstate\_\_**(*state*)

Set the state of the job

**Parameters**

**state** (*dict*) –

**static cast\_to\_type**(*value*, *datatypes*)

Try to cast value to one of the given datatypes. Will try all the datatypes in order.

**Parameters**

**datatypes** (*tuple*) – Possible datatypes to cast to

**Return type**

*DataType*

**Returns**

casted value

**clean**()

**collect\_provenance**()

Collect the provenance for this job.

**property commandfile:** *Path*

The path of the command pickle

**Return type**

*Path*

**property commandurl**

The url of the command pickle

**create\_payload**()

Create the payload for this object based on all the input/output arguments

**Returns**

the payload

**Return type**

*dict*

**ensure\_tmp\_dir**()

**execute()**

Execute this job

**Returns**

The result of the execution

**Return type**

InterFaceResult

**property extrainfofile: Path**

The path where the extra job info document is saved

**Return type**

Path

**property extrainfourl**

The url where the extra job info document is saved

**classmethod fill\_output\_argument(*output\_spec, cardinality, desired\_type, requested, tmpurl*)**

This is an abstract class method. The method should take the `argument_dict` generated from calling `self.get_argument_dict()` and turn it into a list of commandline arguments that represent this Input/Output.

**Parameters**

- **cardinality** (*int*) – the cardinality for this output (can be non for automatic outputs)
- **desired\_type** (*DataType*) – the desired datatype for this output
- **requested** (*bool*) – flag to indicate that the output is requested by Fastr

**Returns**

the values for this output

**Return type**

list

**property fullid**

The full id of the job

**get\_deferred(*output\_id, cardinality\_nr, sample\_id=None*)**

Get a deferred pointing to a specific output value in the Job

**Parameters**

- **output\_id** (*str*) – the output to select from
- **cardinality\_nr** (*int*) – the index of the cardinality
- **sample\_id** (*str*) – the sample id to select (optional)

**Returns**

The deferred

**get\_output\_datatype(*output\_id*)**

Get the datatype for a specific output

**Parameters**

**output\_id** (*str*) – the id of the output to get the datatype for

**Returns**

the requested datatype

**Return type**

tuple

### **get\_result()**

Get the result of the job if it is available. Load the output file if found and check if the job matches the current object. If so, load and return the result.

#### **Returns**

Job after execution or None if not available

#### **Return type**

*Job* | None

### **classmethod get\_value(value)**

Get a value

#### **Parameters**

- **value** – the url of the value
- **datatype** – datatype of the value

#### **Returns**

the retrieved value

### **hash\_inputs()**

Create hashes for all input values and store them in the info store

### **hash\_results()**

Create hashes of all output values and store them in the info store

### **property id**

The id of this job

### **property logfile: Path**

The path of the result pickle

#### **Return type**

*Path*

### **property logurl**

The url of the result pickle

### **property provfile: Path**

The path where the prov document is saved

#### **Return type**

*Path*

### **property provurl**

The url where the prov document is saved

### **property resources**

The compute resources required for this job

### **property status**

The status of the job

### **property stderrfile: Path**

The path where the stderr text is saved

#### **Return type**

*Path*



**property stderrurl**

The url where the stderr text is saved

**property stdoutfile:** `Path`

The path where the stdout text is saved

**Return type**

`Path`

**property stdouturl**

The url where the stdout text is saved

**property tmpdir:** `Path`

Path of tmpdir for the job

**Return type**

`Path`

**property tmpurl**

The URL of the tmpdir to use

**property tool****classmethod translate\_argument**(*value*)

Translate an argument from a URL to an actual path.

**Parameters**

- **value** – value to translate
- **datatype** – the datatype of the value

**Returns**

the translated value

**static translate\_output\_results**(*value, datatypes, mountpoint=None*)

Translate the results for on Output

**Parameters**

- **value** – the results value for the output
- **datatypes** (*tuple*) – tuple of possible datatypes for the output
- **preferred\_type** – the preferred datatype of the output

**Returns**

the update value for the result

**translate\_results**(*result*)

Translate the results of an interface (using paths etc) to the proper form using URI's instead.

**Parameters**

**result** (*dict*) – the result data of an interface

**Returns**

the translated result

**Return type**

`dict`

**validate\_results**(*payload*)

Validate the results of the Job

**Returns**

flag indicating the results are complete and valid

**write**()

**class** fastr.execution.job.JobCleanupLevel(*value*)

Bases: [Enum](#)

The cleanup level for Jobs that are finished.

**\_\_module\_\_** = 'fastr.execution.job'

**all** = 'all'

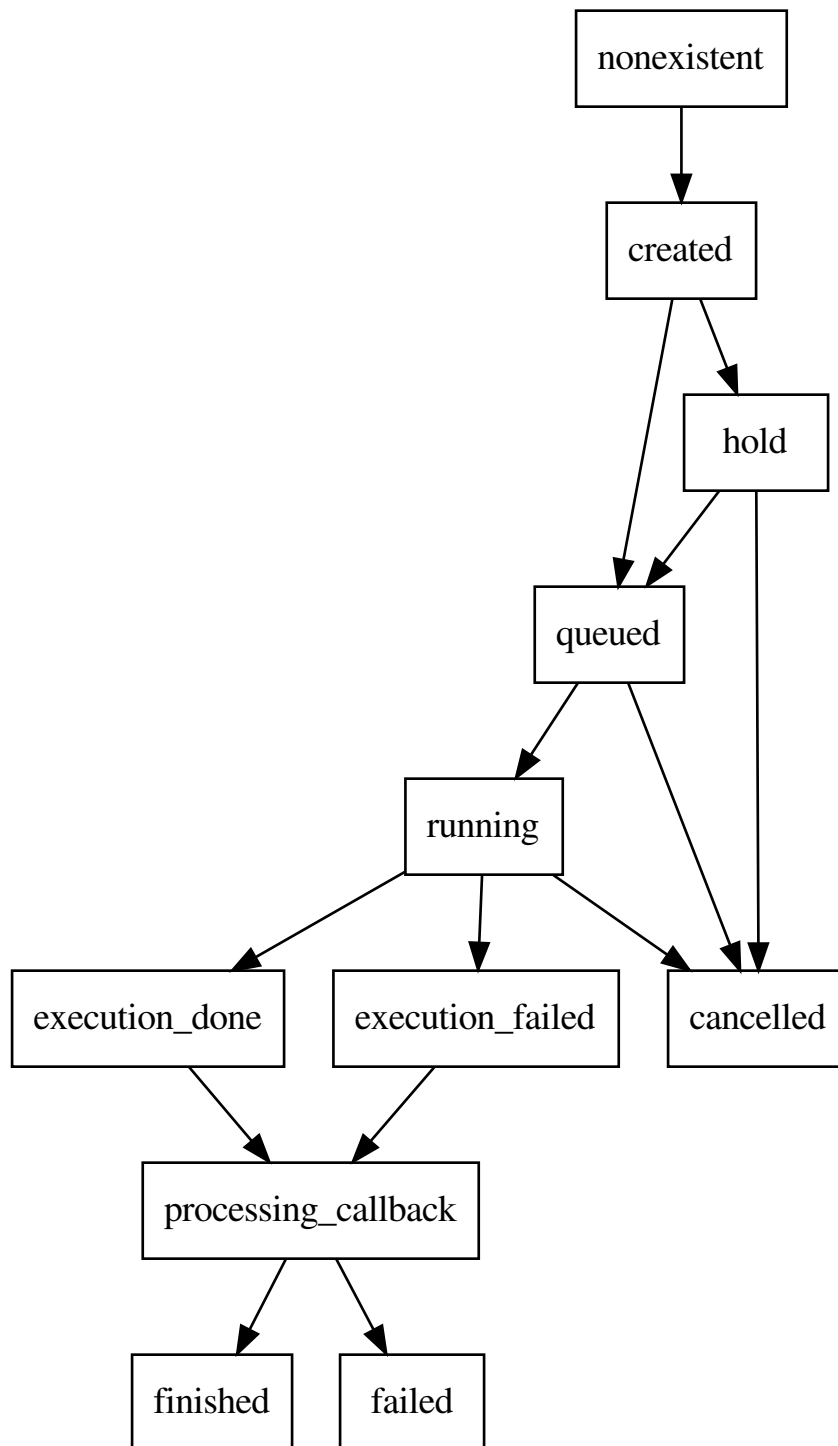
**no\_cleanup** = 'no\_cleanup'

**non\_failed** = 'non\_failed'

**class** fastr.execution.job.JobState(*value*)

Bases: [Enum](#)

The possible states a Job can be in. An overview of the states and the advised transitions are depicted in the following figure:



`__init__(_, stage, error)`

```
__module__ = 'fastr.execution.job'
cancelled = ('cancelled', 'done', True)
created = ('created', 'idle', False)
property done
execution_done = ('execution_done', 'in_progress', False)
execution_failed = ('execution_failed', 'in_progress', True)
execution_skipped = ('execution_skipped', 'in_progress', True)
failed = ('failed', 'done', True)
finished = ('finished', 'done', False)
hold = ('hold', 'idle', False)
property idle
property in_progress
nonexistent = ('nonexistent', 'idle', False)
processing_callback = ('processing_callback', 'in_progress', False)
queued = ('queued', 'idle', False)
running = ('running', 'in_progress', False)
class fastr.execution.job.SinkJob(node, sample_id, sample_index, input_arguments, output_arguments,
                                   hold_jobs=None, substitutions=None, preferred_types=None)
```

Bases: [Job](#)

Special SinkJob for the Sink

**\_\_getstate\_\_()**

Get the state of the job

**Returns**

job state

**Return type**

[dict](#)

**\_\_init\_\_**(node, sample\_id, sample\_index, input\_arguments, output\_arguments, hold\_jobs=None, substitutions=None, preferred\_types=None)

Create a job

**Parameters**

- **node** ([fastr.planning.node.Node](#)) – the node the job is based on
- **sample\_id** – the id of the sample
- **sample\_index** – the index of the sample
- **input\_arguments** – the argument list
- **output\_arguments** – the argument list

- **hold\_jobs** – the jobs on which this jobs depend
- **preferred\_types** – The list of preferred types to use

**Returns**

**\_\_module\_\_** = 'fastr.execution.job'

**\_\_repr\_\_()**

String representation for the SinkJob

**\_\_setstate\_\_**(state)

Set the state of the job

**Parameters**

**state** (*dict*) –

**create\_payload()**

Create the payload for this object based on all the input/output arguments

**Returns**

the payload

**Return type**

*dict*

**get\_result()**

Get the result of the job if it is available. Load the output file if found and check if the job matches the current object. If so, load and return the result.

**Returns**

Job after execution

**hash\_inputs()**

Create hashes for all input values and store them in the info store

**property id**

The id of this job

**substitute**(value, datatype=None)

Substitute the special fields that can be used in a SinkJob.

**Parameters**

- **value** (*str*) – the value to substitute fields in
- **datatype** (*BaseDataType*) – the datatype for the value

**Returns**

string with substitutions performed

**Return type**

*str*

**property tmpurl**

The URL of the tmpdir to use

**validate\_results**(payload)

Validate the results of the SinkJob

**Returns**

flag indicating the results are complete and valid

**class** `fastr.execution.job.SourceJob(datatype, **kwargs)`

Bases: `Job`

Special SourceJob for the Source

**\_\_getstate\_\_()**

Get the state of the job

**Returns**

job state

**Return type**

`dict`

**\_\_init\_\_**(datatype, \*\*kwargs)

Create a job

**Parameters**

- **node** (`fastr.planning.node.Node`) – the node the job is based on
- **sample\_id** – the id of the sample
- **sample\_index** – the index of the sample
- **input\_arguments** – the argument list
- **output\_arguments** – the argument list
- **hold\_jobs** – the jobs on which this jobs depend
- **preferred\_types** – The list of preferred types to use

**Returns**

**\_\_module\_\_** = 'fastr.execution.job'

**\_\_repr\_\_()**

String representation for the SourceJob

**\_\_setstate\_\_**(state)

Set the state of the job

**Parameters**

**state** (`dict`) –

**collect\_provenance()**

Collect the provenance for this job

**create\_payload()**

Create the payload for this object based on all the input/output arguments

**Returns**

the payload

**Return type**

`dict`

**get\_output\_datatype**(output\_id)

Get the datatype for a specific output

**Parameters**

**output\_id** (`str`) – the id of the output to get the datatype for

**Returns**

the requested datatype

**Return type***BaseDataType***hash\_inputs()**

Create hashes for all input values and store them in the info store

**validate\_results(payload)**

Validate the results of the Job

**Returns**

flag indicating the results are complete and valid

**linkrun Module**

The link module contain the Link class. This class represents the links in a network. These links lead from an output (BaseOutput) to an input (BaseInput) and indicate the desired data flow. Links are smart objects, in the sense that when you set their start or end point, they register themselves with the Input and Output. They do all the book keeping, so as long as you only set the source and target of the Link, the link should be valid.

**Warning:** Don't mess with the Link, Input and Output internals from other places. There will be a huge chances of breaking the network functionality!

**class** fastr.execution.linkrun.LinkRun(link, parent=None)

Bases: Updateable, Serializable

Class for linking outputs (*BaseOutput*) to inputs (*BaseInput*)

Examples:

```
>>> import fastr
>>> network = fastr.Network()
>>> link1 = network.create_link( n1.ouputs['out1'], n2.inputs['in2'] )

link2 = Link()
link2.source = n1.ouputs['out1']
link2.target = n2.inputs['in2']
```

**\_\_abstractmethods\_\_** = frozenset({})**\_\_dataschemafile\_\_** = 'Link.schema.json'**\_\_eq\_\_**(other)

Test for equality between two Links

**Parameters****other** (*LinkRun*) – object to test against**Returns**

True for equality, False otherwise

**Return type***bool*

**\_\_getitem\_\_**(*index*)

Get a an item for this Link. The item will be retrieved from the connected output, but a diverging or converging flow can change the number of samples/cardinality.

**Parameters**

**index** (*SampleIndex*) – index of the item to retrieve

**Returns**

the requested item

**Return type**

*SampleItem*

**Raises**

*FastrIndexError* – if the index length does not match the number dimensions in the source data (after collapsing/expanding)

**\_\_getstate\_\_**()

Retrieve the state of the Link

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = *None*

**\_\_init\_\_**(*link, parent=None*)

Create a new Link in a Network.

**Parameters**

- **link** (*Link*) – the base link
- **parent** (*Network* or *None*) – the parent network, if *None* is given the *fastr.current\_network* is assumed to be the parent

**Returns**

newly created *LinkRun*

**Raises**

- *FastrValueError* – if parent is not given and *fastr.current\_network* is not set
- *FastrValueError* – if the source output is not in the same network as the Link
- *FastrValueError* – if the target input is not in the same network as the Link

**\_\_module\_\_** = *'fastr.execution.linkrun'*

**\_\_repr\_\_**()

Get a string representation for the Link

**Returns**

the string representation

**Return type**

*str*

**\_\_setstate\_\_**(*state*)

Set the state of the Link by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with



**Returns**

None

**Raises***FastrValueError* – if the parent network and *fastr.current\_network* are not set**cardinality**(*index=None*)

Cardinality for a Link is given by source Output and the collapse/expand settings

**Parameters****key** (*SampleIndex*) – key for a specific sample (can be only a sample index!)**Returns**

the cardinality

**Return type**

int, sympy.Symbol

**Raises***FastrIndexError* – if the index length does not match the number of dimension in the data**property collapse**

The converging dimensions of this link. Collapsing changes some dimensions of sample lists into cardinality, reshaping the data.

Collapse can be set to a tuple or an int/str, in which case it will be automatically wrapped in a tuple. The int will be seen as indices of the dimensions to collapse. The str will be seen as the name of the dimensions over which to collapse.

**Raises***FastrTypeError* – if assigning a collapse value of a wrong type**property collapse\_indexes**

The converging dimensions of this link as integers. Dimension names are replaced with the corresponding int.

Collapsing changes some dimensions of sample lists into cardinality, reshaping the data

**classmethod createobj**(*state, network=None*)

Create object function for Link

**Parameters**

- **cls** – The class to create
- **state** – The state to use to create the Link
- **network** – the parent Network

**Returns**

newly created Link

**destroy**()

The destroy function of a link removes all default references to a link. This means the references in the network, input and output connected to this link. If there is no references in other places in the code, it will destroy the link (reference count dropping to zero).

This function is called when a source for an input is set to another value and the link becomes disconnected. This makes sure there is no dangling links.

**property dimensions**

The dimensions of the data delivered by the link. This can be different from the source dimensions because the link can make data collapse or expand.

**property expand**

Flag indicating that the link will expand the cardinality into a new sample dimension to be created.

**property fullid**

The full defining ID for the Input

**property parent**

The Network to which this Link belongs.

**property size**

The size of the data delivered by the link. This can be different from the source size because the link can make data collapse or expand.

**property source**

The source *BaseOutput* of the Link. Setting the source will automatically register the Link with the source BaseOutput. Updating source will also make sure the Link is unregistered with the previous source.

**Raises**

*FastrTypeError* – if assigning a non *BaseOutput*

**property status**

**property target**

The target *BaseInput* of the Link. Setting the target will automatically register the Link with the target BaseInput. Updating target will also make sure the Link is unregistered with the previous target.

**Raises**

*FastrTypeError* – if assigning a non *BaseInput*

## macronoderun Module

**class** `fastr.execution.macronoderun.MacroNodeRun(node, parent)`

Bases: *NodeRun*

MacroNodeRun encapsulates an entire network in a single node.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_getstate\_\_()**

Retrieve the state of the MacroNodeRun

**Returns**

the state of the object

**Rtype dict**

**\_\_init\_\_**(*node*, *parent*)

**Parameters**

**network** (`fastr.planning.network.Network`) – network to create macronode for

**\_\_module\_\_** = `'fastr.execution.macronoderun'`

**\_\_setstate\_\_**(*state*)

Set the state of the NodeRun by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**execute**()

Execute the node and create the jobs that need to run

**Returns**

list of jobs to run

**Return type**

list of *Jobs*

**get\_output\_info**(*output*)

**property network\_run**

## networkanalyzer Module

Module that defines the NetworkAnalyzer and holds the reference implementation.

**class** fastr.execution.networkanalyzer.DefaultNetworkAnalyzer

Bases: *NetworkAnalyzer*

Default implementation of the NetworkAnalyzer.

**\_\_module\_\_** = 'fastr.execution.networkanalyzer'

**analyze\_network**(*network, chunk*)

Analyze a chunk of a Network. Simply process the Nodes in the chunk sequentially.

**Parameters**

- **network** – Network corresponding with the chunk
- **chunk** – The chunk of the network to analyze

**class** fastr.execution.networkanalyzer.NetworkAnalyzer

Bases: *object*

Base class for NetworkAnalyzers

```
__dict__ = mappingproxy({'__module__': 'fastr.execution.networkanalyzer',
'__doc__': '\n Base class for NetworkAnalyzers\n ', 'analyze_network': <function
NetworkAnalyzer.analyze_network>, '__dict__': <attribute '__dict__' of
'NetworkAnalyzer' objects>, '__weakref__': <attribute '__weakref__' of
'NetworkAnalyzer' objects>, '__annotations__': {}})
```

**\_\_module\_\_** = 'fastr.execution.networkanalyzer'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**abstract analyze\_network**(*network*, *chunk*)

Analyze a chunk of a Network.

**Parameters**

- **network** – Network corresponding with the chunk
- **chunk** – The chunk of the network to analyze

## networkchunker Module

This module contains the NetworkChunker class and its default implementation the DefaultNetworkChunker

**class** fastr.execution.networkchunker.DefaultNetworkChunker

Bases: *NetworkChunker*

The default implementation of the NetworkChunker. It tries to create as large as possible chunks so the execution blocks as little as possible.

**\_\_init\_\_**()

**\_\_module\_\_** = 'fastr.execution.networkchunker'

**chunk\_network**(*network*)

Create a list of Network chunks that can be pre-analyzed completely. Each chunk needs to be executed before the next can be analyzed and executed.

The returned chunks are (at the moment) in the format of a tuple (start, nodes) which are both tuples. The tuple contain the nodes where to start execution (should ready if previous chunks are done) and all nodes of the chunk respectively.

**Parameters**

**network** – Network to split into chunks

**Returns**

tuple containing chunks

**class** fastr.execution.networkchunker.NetworkChunker

Bases: *object*

The base class for NetworkChunkers. A Network chunker is a class that takes a Network and produces a list of chunks that can each be analyzed and executed in one go.

```
__dict__ = mappingproxy({'__module__': 'fastr.execution.networkchunker', '__doc__':  
'\n The base class for NetworkChunkers. A Network chunker is a class that takes\n a  
Network and produces a list of chunks that can each be analyzed and\n executed in  
one go.\n ', 'chunk_network': <function NetworkChunker.chunk_network>,  
'__dict__': <attribute '__dict__' of 'NetworkChunker' objects>, '__weakref__':  
<attribute '__weakref__' of 'NetworkChunker' objects>, '__annotations__': {}})
```

**\_\_module\_\_** = 'fastr.execution.networkchunker'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**abstract chunk\_network**(*network*)

Create a list of Network chunks that can be pre-analyzed completely. Each chunk needs to be executed before the next can be analyzed and executed.

**Parameters****network** – Network to split into chunks**Returns**

list containing chunks

**networkrun Module**

Network module containing Network facilitators and analysers.

**class** fastr.execution.networkrun.**NetworkRun**(*network*)

Bases: Serializable

The Network class represents a workflow. This includes all Nodes (including ConstantNodes, SourceNodes and Sinks) and Links.

**NETWORK\_DUMP\_FILE\_NAME** = '\_\_fastr\_network\_\_.json'

**SINK\_DUMP\_FILE\_NAME** = '\_\_sink\_data\_\_.json'

**SOURCE\_DUMP\_FILE\_NAME** = '\_\_source\_data\_\_.pickle.gz'

**\_\_bool\_\_**()

A network run is True if it finish running successfully and False otherwise

**\_\_eq\_\_**(*other*)

Compare two Networks and see if they are equal.

**Parameters****other** (*Network*) –**Returns**

flag indicating that the Networks are the same

**Return type***bool*

**\_\_getitem\_\_**(*item*)

Get an item by its fullid. The fullid can point to a link, node, input, output or even subinput/suboutput.

**Parameters****item** (*str*, *unicode*) – fullid of the item to retrieve**Returns**

the requested item

**\_\_getstate\_\_**()

Retrieve the state of the Network

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = None

**\_\_init\_\_**(*network*)

Create a new, empty Network

**Parameters****name** (*str*) – name of the Network

**Returns**

newly created Network

**Raises**

**OSError** – if the tmp mount in the config is not a writable directory

**\_\_module\_\_** = 'fastr.execution.networkrun'

**\_\_ne\_\_**(*other*)

Tests for non-equality, this is the negated version **\_\_eq\_\_**

**\_\_repr\_\_**()

Return repr(self).

**\_\_setstate\_\_**(*state*)

Set the state of the Network by the given state. This completely overwrites the old state!

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**abort**(*signal\_code=None, current\_frame=None*)

**check\_id**(*id\_*)

Check if an id for an object is valid and unused in the Network. The method will always returns True if it does not raise an exception.

**Parameters**

**id** (*str*) – the id to check

**Returns**

True

**Raises**

- **FastrValueError** – if the id is not correctly formatted
- **FastrValueError** – if the id is already in use

**property constantlist**

**execute**(*sourcedata, sinkdata, execution\_plugin=None, tmpdir=None, cluster\_queue=None, timestamp=None, tracking\_id=None*)

Execute the Network with the given data. This will analyze the Network, create jobs and send them to the execution backend of the system.

**Parameters**

- **sourcedata** (*dict*) – dictionary containing all data for the sources
- **sinkdata** (*dict*) – dictionary containing directives for the sinks
- **execution\_plugin** (*str*) – the execution plugin to use (None will use the config value)

**Raises**

- **FastrKeyError** – if a source has not corresponding key in sourcedata
- **FastrKeyError** – if a sink has not corresponding key in sinkdata

**execution\_finished**()

**property fullid**

The fullid of the Network

**generate\_jobs()****property global\_id**

The global id of the Network, this is different for networks used in macronodes, as they still have parents.

**property id**

The id of the Network. This is a read only property.

**job\_finished(*job*)**

Call-back handler for when a job is finished. Will collect the results and handle blocking jobs. This function is automatically called when the execution plugin finished a job.

**Parameters**

**job** (*Job*) – the job that finished

**property long\_id****property network****property nodegroups**

Give an overview of the nodegroups in the network

**register\_signals()**

Register handles to handle SIGINT and SIGTERM handlers to gracefully shut down the execution :return:

**set\_data(*sourcedata*, *sinkdata*)****property sinklist****property sourcelist****unregister\_signals()**

Unregister the signal handlers (set to default). Sending these signals twice will result that the second time the default handler is used.

**noderun Module**

A module to maintain a run of a network node.

**class fastr.execution.noderun.NodeRun(*node*, *parent*)**

Bases: *BaseNodeRun*

The class encapsulating a node in the network. The node is responsible for setting and checking inputs and outputs based on the description provided by a tool instance.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_dataschemafile\_\_** = 'NodeRun.schema.json'

**\_\_eq\_\_**(*other*)

Compare two Node instances with each other. This function ignores the parent and update status, but tests rest of the dict for equality. equality

**Parameters**

**other** (*NodeRun*) – the other instances to compare to

**Returns**

True if equal, False otherwise

**\_\_getstate\_\_()**

Retrieve the state of the NodeRun

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = None

**\_\_init\_\_**(*node*, *parent*)

Instantiate a node.

**Parameters**

- **node** (*Tool*) – The node to base the noderun on
- **parent** (*Network*) – the parent network of the node

**Returns**

the newly created NodeRun

**\_\_module\_\_** = 'fastr.execution.noderun'

**\_\_repr\_\_**()

Get a string representation for the NodeRun

**Returns**

the string representation

**Return type**

*str*

**\_\_setstate\_\_**(*state*)

Set the state of the NodeRun by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**\_\_str\_\_**()

Get a string version for the NodeRun

**Returns**

the string version

**Return type**

*str*

**property blocking**

Indicate that the results of this NodeRun cannot be determined without first executing the NodeRun, causing a blockage in the creation of jobs. A blocking Nodes causes the Chunk borders.

**create\_job**(*sample\_id*, *sample\_index*, *job\_data*, *job\_dependencies*, *status*, *\*\*kwargs*)

Create a job based on the sample id, job data and job dependencies.

**Parameters**



- **sample\_id** (`SampleId`) – the id of the corresponding sample
- **sample\_index** (`SampleIndex`) – the index of the corresponding sample
- **job\_data** (*dict*) – dictionary containing all input data for the job
- **job\_dependencies** – other jobs that need to finish before this job can run

**Returns**

the created job

**Return type**

*Job*

**classmethod** `createobj`(*state*, *network=None*)

Create object function for generic objects

**Parameters**

- **cls** – The class to create
- **state** – The state to use to create the Link

**Returns**

newly created Link

**property** `dimnames`

Names of the dimensions in the NodeRun output. These will be reflected in the SampleIdList of this NodeRun.

**execute**()

Execute the node and create the jobs that need to run

**Returns**

list of jobs to run

**Return type**

list of *Jobs*

**find\_source\_index**(*target\_index*, *target*, *source*)

**property** `fullid`

The full defining ID for the NodeRun inside the network

**get\_sourced\_nodes**()

A list of all Nodes connected as sources to this NodeRun

**Returns**

list of all nodes that are connected to an input of this node

**property** `global_id`

The global defining ID for the Node from the main network (goes out of macro nodes to root network)

**property** `id`

The id of the NodeRun

**property** `input_groups`

**A list of input groups for this NodeRun. An input group is `InputGroup` object filled according to the NodeRun**

**property** `listeners`

All the listeners requesting output of this node, this means the listeners of all Outputs and SubOutputs

**property merge\_dimensions**

**property name**

Name of the Tool the NodeRun was based on. In case a Toolless NodeRun was used the class name is given.

**property outputs\_size**

Size of the outputs in this NodeRun

**property parent**

The parent network of this node.

**property resources**

Number of cores required for the execution of this NodeRun

**set\_result**(*job, failed\_annotation*)

Incorporate result of a job into the NodeRun.

**Parameters**

- **job** (*Type*) – job of which the result to store
- **failed\_annotation** – A set of annotations, None if no errors else containing a tuple describing the errors

**property status**

**property tool**

**update\_input\_groups**()

Update all input groups in this node

## sinknoderun Module

**class** `fastr.execution.sinknoderun.SinkNodeRun(node, parent)`

Bases: `NodeRun`

Class which handles where the output goes. This can be any kind of file, e.g. image files, textfiles, config files, etc.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_dataschemafile\_\_** = `'SinkNodeRun.schema.json'`

**\_\_getstate\_\_**()

Retrieve the state of the NodeRun

**Returns**

the state of the object

**Rtype dict**

**\_\_init\_\_**(*node, parent*)

Instantiation of the SinkNodeRun.

**Parameters**

- **node** (`fastr.planning.node.Node`) – The Node that this Run is based on.
- **parent** (`NetworkRun`) – The NetworkRun that this NodeRun belongs to

**Returns**

newly created sink node run

**\_\_module\_\_** = 'fastr.execution.sinknoderun'

**\_\_setstate\_\_**(*state*)

Set the state of the NodeRun by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**create\_job**(*sample\_id*, *sample\_index*, *job\_data*, *job\_dependencies*, *status*, *\*\*kwargs*)

Create a job for a sink based on the sample id, job data and job dependencies.

**Parameters**

- **sample\_id** (*SampleId*) – the id of the corresponding sample
- **job\_data** (*dict*) – dictionary containing all input data for the job
- **job\_dependencies** – other jobs that need to finish before this job can run

**Returns**

the created job

**Return type**

*Job*

**property datatype**

The datatype of the data this sink can store.

**property input**

The default input of the sink NodeRun

**set\_data**(*data*)

Set the targets of this sink node.

**Parameters**

**data** (*dict* or *list of urls*) – the targets rules for where to write the data

The target rules can include a few fields that can be filled out:

field	description
sample_id	the sample id of the sample written in string form
cardinality	the cardinality of the sample written
ext	the extension of the datatype of the written data, including the .
extension	the extension of the datatype of the written data, excluding the .
network	the id of the network the sink is part of
node	the id of the node of the sink
timestamp	the iso formatted datetime the network execution started
uuid	the uuid of the network run (generated using uuid.uuid1)

An example of a valid target could be:

```
>>> target = 'vfs://output_mnt/some/path/image_{sample_id}_{cardinality}{ext}'
```

---

**Note:** The `{ext}` and `{extension}` are very similar but are both offered. In many cases having a `name.{extension}` will feel like the correct way to do it. However, if you have DataTypes with and without extension that can both be exported by the same sink, this would cause either `name.ext` or `name.` to be generated. In this particular case `name{ext}` can help as it will create either `name.ext` or `name.`

---

---

**Note:** If a datatype has multiple extensions (e.g. `.tiff` and `.tif`) the first extension defined in the extension tuple of the datatype will be used.

---

**set\_result**(*job, failed\_annotation*)

Incorporate result of a sink job into the Network.

**Parameters**

- **job** (*Type*) – job of which the result to store
- **failed\_annotation** (*set*) – A set of annotations, None if no errors else containing a tuple describing the errors

## sourcenoderun Module

**class** `fastr.execution.sourcenoderun.ConstantNodeRun`(*node, parent*)

Bases: `SourceNodeRun`

Class encapsulating one output for which a value can be set. For example used to set a scalar value to the input of a node.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_dataschemafile\_\_** = `'ConstantNodeRun.schema.json'`

**\_\_getstate\_\_**()

Retrieve the state of the ConstantNodeRun

**Returns**

the state of the object

**Rtype dict**

**\_\_init\_\_**(*node, parent*)

Instantiation of the ConstantNodeRun.

**Parameters**

- **datatype** – The datatype of the output.
- **data** – the prefilled data to use.
- **id** – The url pattern.

This class should never be instantiated directly (unless you know what you are doing). Instead create a constant using the network class like shown in the usage example below.

usage example:

```
>>> import fastr
>>> network = fastr.Network()
>>> source = network.create_source(datatype=types['ITKImageFile'], id_='sourceN
↪')
```

or alternatively create a constant node by assigning data to an item in an InputDict:

```
>>> node_a.inputs['in'] = ['some', 'data']
```

which automatically creates and links a ConstantNodeRun to the specified Input

```
__module__ = 'fastr.execution.sourcenoderun'
```

```
__setstate__(state)
```

Set the state of the ConstantNodeRun by the given state.

#### Parameters

**state** (*dict*) – The state to populate the object with

#### Returns

None

#### property data

The data stored in this constant node

```
set_data(data=None, ids=None)
```

Set the data of this constant node in the correct way. This is mainly for compatibility with the parent class SourceNodeRun

#### Parameters

- **data** (*dict* or *list* of *urls*) – the data to use
- **ids** – if data is a list, a list of accompanying ids

```
class fastr.execution.sourcenoderun.SourceNodeRun(node, parent)
```

Bases: [FlowNodeRun](#)

Class providing a connection to data resources. This can be any kind of file, stream, database, etc from which data can be received.

```
__abstractmethods__ = frozenset({})
```

```
__dataschemafile__ = 'SourceNodeRun.schema.json'
```

```
__eq__(other)
```

Compare two Node instances with each other. This function ignores the parent and update status, but tests rest of the dict for equality. equality

#### Parameters

**other** (*NodeRun*) – the other instances to compare to

#### Returns

True if equal, False otherwise

```
__getstate__()
```

Retrieve the state of the SourceNodeRun

#### Returns

the state of the object

**Rtype dict**

**\_\_hash\_\_** = None

**\_\_init\_\_**(*node, parent*)

Instantiation of the SourceNodeRun.

**Parameters**

- **node** (`fastr.planning.node.Node`) – The Node that this Run is based on.
- **parent** (`NetworkRun`) – The NetworkRun that this NodeRun belongs to

**Returns**

newly created sink node run

**\_\_module\_\_** = 'fastr.execution.sourcenoderun'

**\_\_setstate\_\_**(*state*)

Set the state of the SourceNodeRun by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**create\_job**(*sample\_id, sample\_index, job\_data, job\_dependencies, status, \*\*kwargs*)

Create a job based on the sample id, job data and job dependencies.

**Parameters**

- **sample\_id** (`SampleId`) – the id of the corresponding sample
- **sample\_index** (`SampleIndex`) – the index of the corresponding sample
- **job\_data** (*dict*) – dictionary containing all input data for the job
- **job\_dependencies** – other jobs that need to finish before this job can run

**Returns**

the created job

**Return type**

*Job*

**property datatype**

The datatype of the data this source supplies.

**property dimnames**

Names of the dimensions in the SourceNodeRun output. These will be reflected in the SampleIdLists.

**property output**

Shorthand for `self.outputs['output']`

**property outputsize**

The size of output of this SourceNodeRun

**set\_data**(*data, ids=None*)

Set the data of this source node.

**Parameters**

- **data** (*dict, OrderedDict or list of urls*) – the data to use

- **ids** – if data is a list, a list of accompanying ids

**property sourcegroup**

**property valid**

This does nothing. It only overloads the valid method of NodeRun(). The original is intended to check if the inputs are connected to some output. Since this class does not implement inputs, it is skipped.

**helpers Package**

**helpers Package**

```
fastr.helpers.config = # [bool] Flag to enable/disable debugging debug = False # [str]
Directory containing the fastr examples examplesdir = "/home/docs/checkouts/readthedocs.
org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/examples" # [str] The
default execution plugin to use execution_plugin = "ProcessPoolExecution" # [str]
Execution script location executionscript =
"/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/execution/executionscript.py" # [list] Extra configuration
directories to read extra_config_dirs = [ "" ] # [str] Redis url e.g.
redis://localhost:6379 filesynchelper_url = "" # [str] The level of cleanup required,
options: all, no_cleanup, non_failed job_cleanup_level = "no_cleanup" # [bool] Indicate
if default logging settings should log to files or not log_to_file = False # [str]
Directory where the fastr logs will be placed logdir = "/home/docs/.fastr/logs" # [dict]
Python logger config logging_config = {} # [int] The log level to use (as int), INFO is
20, WARNING is 30, etc loglevel = 20 # [str] Type of logging to use logtype = "default" #
[dict] A dictionary containing all mount points in the VFS system mounts = { "tmp":
"/tmp", "examples": "/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/
lib/python3.6/site-packages/fastr/examples", "example_data":
"/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/examples/data", "home": "/home/docs", "fastr_home":
"/home/docs/.fastr" } # [list] Directories to scan for networks networks_path = [
"/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/networks" ] # [list] Directories to scan for plugins
plugins_path = [ "/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins" ] # [list] A list indicating the order
of the preferred types to use. First item is most preferred. preferred_types = [] #
[list] A list of modules in the environmmnet modules that are protected against unloading
protected_modules = [] # [int] Interval in which to report the number of queued jobs
(default is 0, no reporting) queue_report_interval = 0 # [list] The reporting plugins to
use, is a list of all plugins to be activated reporting_plugins = [ "SimpleReport" ] #
[str] Directory containing the fastr system resources resourcesdir =
"/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources" # [str] Directory containing the fastr data schemas
schemadir = "/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/schemas" # [int] The number of source jobs
allowed to run concurrently source_job_limit = 0 # [str] Fastr installation directory
systemdir = "/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr" # [list] Directories to scan for tools tools_path = [
"/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/tools" ] # [list] Directories to scan for datatypes
types_path = [ "/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/datatypes" ] # [str] Fastr user configuration
directory userdir = "/home/docs/.fastr" # [bool] Warning users on import if this is not a
production version of fastr warn_develop = True # [str] The hostname to expose the web
app for web_hostname = "localhost" # [int] The interval in which the job checker will
start to check for stale jobs slurm_job_check_interval = 30 # [str] The slurm partition to
use slurm_partition = "" # [int] Number of workers to use in a process pool
process_pool_worker_number = 1 # [str] The PIM host to report to pim_host = "" # [str]
Username to send to PIM pim_username = "docs" # [float] The interval in which to send
jobs to PIM pim_update_interval = 2.5 # [int] Maximum number of jobs that can be send to
PIM in a single interval pim_batch_size = 100 # [bool] Setup PIM debug mode to send
stdout stderr on job success pim_debug = False # [int] Maximum number of seconds after
the network finished in which PIM tries to synchronize all remaining jobs
pim_finished_timeout = 10
```

Configuration of the fastr system



## checksum Module

This module contains a number of functions for checksumming files and objects

`fastr.helpers.checksum.checksum(filepath, algorithm='md5', hasher=None, chunksize=32768)`

Generate the checksum of a file

### Parameters

- **filepath** (*str*, *list*) – path of the file(s) to checksum
- **algorithm** (*str*) – the algorithm to use
- **hasher** (*\_hashlib.HASH*) – a hasher to continue updating (rather than creating a new one)

### Returns

the checksum

### Return type

*str*

`fastr.helpers.checksum.checksum_directory(directory, algorithm='md5', hasher=None)`

Generate the checksum of an entire directory

### Parameters

- **directory** (*str*) – path of the file(s) to checksum
- **algorithm** (*str*) – the algorithm to use
- **hasher** (*\_hashlib.HASH*) – a hasher to continue updating (rather than creating a new one)

### Returns

the checksum

### Return type

*str*

`fastr.helpers.checksum.hashsum(objects, hasher=None)`

Generate the md5 checksum of (a) python object(s)

### Parameters

- **objects** – the objects to hash
- **hasher** – the hasher to use as a base

### Returns

the hash generated

### Return type

*str*

`fastr.helpers.checksum.md5_checksum(filepath)`

Generate the md5 checksum of a file

### Parameters

**filepath** (*str*, *list*) – path of the file(s) to checksum

### Returns

the checksum

### Return type

*str*

`fastr.helpers.checksum.sha1_checksum(filepath)`

Generate the sha1 checksum of a file

**Parameters**

**filepath** (*str*, *list*) – path of the file(s) to checksum

**Returns**

the checksum

**Return type**

*str*

## classproperty Module

Module containing the code to create class properties.

**class** `fastr.helpers.classproperty.ClassPropertyDescriptor(fget)`

Bases: *object*

A descriptor that can act like a property for a class.

```
__dict__ = mappingproxy({'__module__': 'fastr.helpers.classproperty', '__doc__':  
'\n A descriptor that can act like a property for a class.\n ', '__init__':  
<function ClassPropertyDescriptor.__init__>, '__get__': <function  
ClassPropertyDescriptor.__get__>, '__dict__': <attribute '__dict__' of  
'ClassPropertyDescriptor' objects>, '__weakref__': <attribute '__weakref__' of  
'ClassPropertyDescriptor' objects>, '__annotations__': {}})
```

```
__get__(obj, cls=None)
```

```
__init__(fget)
```

```
__module__ = 'fastr.helpers.classproperty'
```

```
__weakref__
```

list of weak references to the object (if defined)

`fastr.helpers.classproperty.classproperty(func)`

Decorator to create a “class property”

**Parameters**

**func** – the function to wrap

**Returns**

a class property

**Return type**

*ClassPropertyDescriptor*

### clear\_pycs Module

A small tool to wipe all .pyc files from fastr

`fastr.helpers.clear_pycs.dir_list(directory)`

Find all .pyc files

**Parameters**

**directory** (*str*) – directory to search

**Returns**

all .pyc files

**Return type**

*list*

`fastr.helpers.clear_pycs.main()`

Main entry poitn

### configmanager Module

This module defines the Fastr Config class for managing the configuration of Fastr. The config object is stored directly in the fastr top-level module.

**class** `fastr.helpers.configmanager.Config(*configfiles)`

Bases: *object*

Class contain the fastr configuration

```

DEFAULT_FIELDS = {'debug': (<class 'bool'>, False, 'Flag to enable/disable
debugging'), 'examplesdir': (<class 'str'>, '/home/docs/checkouts/readthedocs.org/
user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/examples', 'Directory
containing the fastr examples', '$systemdir/examples'), 'execution_plugin': (<class
'str'>, 'ProcessPoolExecution', 'The default execution plugin to use'),
'executionscript': (<class 'str'>,
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/execution/executionscript.py', 'Execution script location',
'$systemdir/execution/executionscript.py'), 'extra_config_dirs': (<class 'list'>,
[''], 'Extra configuration directories to read'), 'filesynchelper_url': (<class
'str'>, '', 'Redis url e.g. redis://localhost:6379'), 'job_cleanup_level': (<class
'str'>, 'no_cleanup', 'The level of cleanup required, options: all, no_cleanup,
non_failed', 'no_cleanup', <function Config.<lambda>>), 'log_to_file': (<class
'bool'>, False, 'Indicate if default logging settings should log to files or not'),
'logdir': (<class 'str'>, '/home/docs/.fastr/logs', 'Directory where the fastr logs
will be placed', '$userdir/logs'), 'logging_config': (<class 'dict'>, {}, 'Python
logger config'), 'loglevel': (<class 'int'>, 20, 'The log level to use (as int),
INFO is 20, WARNING is 30, etc'), 'logtype': (<class 'str'>, 'default', 'Type of
logging to use'), 'mounts': (<class 'dict'>, {'tmp': '/tmp', 'examples':
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/examples', 'example_data':
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/examples/data', 'home': '/home/docs', 'fastr_home':
'/home/docs/.fastr'}, 'A dictionary containing all mount points in the VFS system',
{'tmp': '$TMPDIR', 'examples': '$systemdir/examples', 'example_data':
'$systemdir/examples/data', 'home': '~/', 'fastr_home': '$FASTRHOME or
~/.fastr'}), 'networks_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/networks'], 'Directories to scan for networks',
['$userdir/networks', '$resourcedir/networks']), 'plugins_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins'], 'Directories to scan for plugins',
['$userdir/plugins', '$resourcedir/plugins']), 'preferred_types': (<class 'list'>,
[], 'A list indicating the order of the preferred types to use. First item is most
preferred.'), 'protected_modules': (<class 'list'>, [], 'A list of modules in the
environmmnet modules that are protected against unloading'), 'queue_report_interval':
(<class 'int'>, 0, 'Interval in which to report the number of queued jobs (default
is 0, no reporting)'), 'reporting_plugins': (<class 'list'>, ['SimpleReport'], 'The
reporting plugins to use, is a list of all plugins to be activated'),
'resourcesdir': (<class 'str'>, '/home/docs/checkouts/readthedocs.org/user_builds/
fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources', 'Directory containing
the fastr system resources', '$systemdir/resources'), 'schemadir': (<class 'str'>,
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/schemas', 'Directory containing the fastr data
schemas', '$systemdir/schemas'), 'source_job_limit': (<class 'int'>, 0, 'The number
of source jobs allowed to run concurrently'), 'systemdir': (<class 'str'>,
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr', 'Fastr installation directory', 'Directory of the top-level
fastr package'), 'tools_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/tools'], 'Directories to scan for tools',
['$userdir/tools', '$resourcedir/tools']), 'types_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/datatypes'], 'Directories to scan for datatypes',
['$userdir/datatypes', '$resourcedir/datatypes']), 'userdir': (<class 'str'>,
'/home/docs/.fastr', 'Fastr user configuration directory', '$FASTRHOME or
~/.fastr'), 'warn_develop': (<class 'bool'>, True, 'Warning developer mode, if set
is not a production version of fastr'), 'web_hostname': (<class 'str'>,
'localhost', 'The hostname to expose the web app for')}}

```



```

__dict__ = mappingproxy({'__module__': 'fastr.helpers.configmanager', '__doc__':
'\n Class contain the fastr configuration\n ', 'DEFAULT_FIELDS': {'logging_config':
(<class 'dict'>, {}, 'Python logger config'), 'extra_config_dirs': (<class 'list'>,
[], 'Extra configuration directories to read'), 'debug': (<class 'bool'>, False,
'Flag to enable/disable debugging'), 'logtype': (<class 'str'>, 'default', 'Type of
logging to use'), 'log_to_file': (<class 'bool'>, False, 'Indicate if default
logging settings should log to files or not'), 'loglevel': (<class 'int'>, 20, 'The
log level to use (as int), INFO is 20, WARNING is 30, etc'), 'systemdir': (<class
'str'>, '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr', 'Fastr installation directory', 'Directory of the
top-level fastr package'), 'userdir': (<class 'str'>, '/home/docs/.fastr', 'Fastr
user configuration directory', '$FASTRHOME or ~/.fastr'), 'logdir': (<class 'str'>,
'/home/docs/.fastr/logs', 'Directory where the fastr logs will be placed',
'$userdir/logs'), 'resourcesdir': (<class 'str'>,
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources', 'Directory containing the fastr system resources',
'$systemdir/resources'), 'examplesdir': (<class 'str'>,
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/examples', 'Directory containing the fastr examples',
'$systemdir/examples'), 'schemadir': (<class 'str'>,
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/schemas', 'Directory containing the fastr data
schemas', '$systemdir/schemas'), 'executionscript': (<class 'str'>,
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/execution/executionscript.py', 'Execution script location',
'$systemdir/execution/executionscript.py'), 'types_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/datatypes'], 'Directories to scan for datatypes',
['$userdir/datatypes', '$resourcesdir/datatypes']), 'tools_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/tools'], 'Directories to scan for tools',
['$userdir/tools', '$resourcesdir/tools']), 'networks_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/networks'], 'Directories to scan for networks',
['$userdir/networks', '$resourcesdir/networks']), 'plugins_path': (<class 'list'>,
['/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins'], 'Directories to scan for plugins',
['$userdir/plugins', '$resourcesdir/plugins']), 'mounts': (<class 'dict'>, {'tmp':
'/tmp', 'examples': '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.
3.1/lib/python3.6/site-packages/fastr/examples', 'example_data':
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/examples/data', 'home': '/home/docs', 'fastr_home':
'/home/docs/.fastr'}, 'A dictionary containing all mount points in the VFS system',
{'tmp': '$TMPDIR', 'examples': '$systemdir/examples', 'example_data':
'$systemdir/examples/data', 'home': '~/', 'fastr_home': '$FASTRHOME or
~/.fastr'}), 'preferred_types': (<class 'list'>, [], 'A list indicating the order
of the preferred types to use. First item is most preferred.'), 'protected_modules':
(<class 'list'>, [], 'A list of modules in the environmmnet modules that are
protected against unloading'), 'execution_plugin': (<class 'str'>,
'ProcessPoolExecution', 'The default execution plugin to use'), 'reporting_plugins':
(<class 'list'>, ['SimpleReport'], 'The reporting plugins to use, is a list of all
plugins to be activated'), 'web_hostname': (<class 'str'>, 'localhost', 'The
hostname to expose the web app for'), 'warn_develop': (<class 'bool'>, True,
'Warning users on import if this is not a production version of fastr'),
'source_job_limit': (<class 'int'>, 0, 'The number of source jobs allowed to run
concurrently'), 'job_cleanup_level': (<class 'str'>, 'no_cleanup', 'The level of
cleanup required, options: all, no_cleanup, job_cleanup'), 'redis_url': (<class
'str'>, '', 'Redis url e.g. redis://localhost:6379'), 'queue_report_interval': (<class
'int'>, 0, 'Interval in which to report the number of queued jobs (default is 0, no reporting)')},

```

```
__init__(*configfiles)  
__module__ = 'fastr.helpers.configmanager'  
__repr__()  
    Return repr(self).  
__weakref__  
    list of weak references to the object (if defined)  
property debug  
property examplesdir  
property execution_plugin  
property executionscript  
property extra_config_dirs  
property filesynchelper_url  
get_field(item)  
property job_cleanup_level  
property log_to_file  
property logdir  
property logging_config  
property loglevel  
property logtype  
property mounts  
property networks_path  
property pim_batch_size  
property pim_debug  
property pim_finished_timeout  
property pim_host  
property pim_update_interval  
property pim_username  
property plugins_path  
property preferred_types  
property process_pool_worker_number  
property protected_modules
```

**property queue\_report\_interval**

**read\_config**(*filename*)

Read a configuration and update the configuration object accordingly

**Parameters**

**filename** – the configuration file to read

**read\_config\_files**

Trace of the config files read by this object

**read\_config\_string**(*value*)

**register\_fields**(*fields\_spec*)

Register extra fields to the configuration manager.

**property reporting\_plugins**

**property resourcesdir**

**property schemadir**

**set\_field**(*item, value*)

**property slurm\_job\_check\_interval**

**property slurm\_partition**

**property source\_job\_limit**

**property systemdir**

**property tools\_path**

**property types\_path**

**property userdir**

**property warn\_develop**

**property web\_hostname**

**web\_url**()

Construct a fqdn from the web['hostname'] and web['port'] settings. :return: FQDN :rtype: str

**class** fastr.helpers.configmanager.**EmptyDefault**(*data=None*)

Bases: `object`

Empty defaultdict.

**\_\_add\_\_**(*right*)

**\_\_delitem\_\_**(*key*)



```
__dict__ = mappingproxy({'__module__': 'fastr.helpers.configmanager', '__doc__': '
Empty defaultdict. ', '__init__': <function EmptyDefault.__init__>, '__iadd__':
<function EmptyDefault.__iadd__>, '__add__': <function EmptyDefault.__add__>,
'__radd__': <function EmptyDefault.__radd__>, 'append': <function
EmptyDefault.append>, 'prepend': <function EmptyDefault.prepend>, 'extend':
<function EmptyDefault.extend>, 'update': <function EmptyDefault.update>,
'merge_default': <function EmptyDefault.merge_default>, '__getitem__': <function
EmptyDefault.__getitem__>, '__setitem__': <function EmptyDefault.__setitem__>,
'__delitem__': <function EmptyDefault.__delitem__>, 'aslist': <function
EmptyDefault.aslist>, 'asdict': <function EmptyDefault.asdict>, '__dict__':
<attribute '__dict__' of 'EmptyDefault' objects>, '__weakref__': <attribute
'__weakref__' of 'EmptyDefault' objects>, '__annotations__': {}})
```

**\_\_getitem\_\_**(*item*)

**\_\_iadd\_\_**(*right*)

**\_\_init\_\_**(*data=None*)

**\_\_module\_\_** = 'fastr.helpers.configmanager'

**\_\_radd\_\_**(*other*)

**\_\_setitem\_\_**(*key, value*)

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**append**(*value*)

**asdict**()

**aslist**()

**extend**(*other*)

**merge\_default**(*field\_spec*)

Merge the default into this EmptyDefault given the field spec :param field\_spec: Field specification :return: Merged value

**prepend**(*value*)

**update**(*other*)

**class** fastr.helpers.configmanager.FastrLogRecordFilter(*name=""*)

Bases: [Filter](#)

**\_\_module\_\_** = 'fastr.helpers.configmanager'

**filter**(*record*)

Determine if the specified record is to be logged.

Is the specified record to be logged? Returns 0 for no, nonzero for yes. If deemed appropriate, the record may be modified in-place.

**Return type**

[bool](#)

## events Module

**class** `fastr.helpers.events.EventType(value)`

Bases: `Enum`

An enumeration.

`__module__` = `'fastr.helpers.events'`

`job_updated` = `'job_updated'`

`log_record_emitted` = `'log_record_emitted'`

`run_finished` = `'run_finished'`

`run_started` = `'run_started'`

**class** `fastr.helpers.events.FastrLogEventHandler(level=0)`

Bases: `Handler`

Logging handler that sends the log records into the event system

`__module__` = `'fastr.helpers.events'`

**emit**(*record*)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a `NotImplementedError`.

`fastr.helpers.events.emit_event(event_type, data)`

Emit an event to all listeners :type event\_type: `EventType` :param event\_type: The type of event to emit :param data: The data object to send along

`fastr.helpers.events.register_listener(event_type, function)`

Register a listeners to a specific event type

### Parameters

- **event\_type** (`EventType`) – The EventType to listen on
- **function** (`Callable[[object], None]`) – The callable that will be called on each event

`fastr.helpers.events.remove_listener(event_type, function)`

Remove a listener from a type of event

### Parameters

- **event\_type** (`EventType`) – The event type to remove the listeners from
- **function** (`Callable[[object], None]`) – The function to remove

## filesynchelper Module

Some helper functions that aid with NFS file sync issues.

**class** fastr.helpers.filesynchelper.FileSyncHelper

Bases: `object`

```
__dict__ = mappingproxy({'__module__': 'fastr.helpers.filesynchelper',
'_namespace': 'filesynchelper', '_redis': None, '__init__': <function
FileSyncHelper.__init__>, 'job_finished': <function FileSyncHelper.job_finished>,
'wait_for_job': <function FileSyncHelper.wait_for_job>, 'wait_for_pickle':
<function FileSyncHelper.wait_for_pickle>, 'store': <function
FileSyncHelper.store>, 'load': <function FileSyncHelper.load>,
'_generate_key_for_string': <function FileSyncHelper._generate_key_for_string>,
'_generate_hash_from_string': <function FileSyncHelper._generate_hash_from_string>,
'make_file_promise': <function FileSyncHelper.make_file_promise>,
'has_file_promise': <function FileSyncHelper.has_file_promise>, 'wait_for_vfs_url':
<function FileSyncHelper.wait_for_vfs_url>, 'wait_for_file': <function
FileSyncHelper.wait_for_file>, '_get_suburl_hashes': <function
FileSyncHelper._get_suburl_hashes>, '_glob_dir': <function
FileSyncHelper._glob_dir>, '_wait_for_file_and_suburls': <function
FileSyncHelper._wait_for_file_and_suburls>, '__dict__': <attribute '__dict__' of
'FileSyncHelper' objects>, '__weakref__': <attribute '__weakref__' of
'FileSyncHelper' objects>, '__doc__': None, '__annotations__': {}})
```

`__init__()`

`__module__ = 'fastr.helpers.filesynchelper'`

`__weakref__`

list of weak references to the object (if defined)

`has_file_promise(url)`

`job_finished(jobfile)`

`load(url)`

`make_file_promise(url)`

`store(url, data)`

`wait_for_file(path, timeout=300)`

`wait_for_job(jobfile)`

`wait_for_pickle(url, timeout=300)`

`wait_for_vfs_url(vfs_url, timeout=300)`

`fastr.helpers.filesynchelper.filesynchelper_enabled()`

## iohelpers Module

```
fastr.helpers.iohelpers.link_or_copy(source, destination)
fastr.helpers.iohelpers.load_gpickle(path, retry_scheme=None)
fastr.helpers.iohelpers.load_json(path)
fastr.helpers.iohelpers.save_gpickle(path, data)
fastr.helpers.iohelpers.save_json(path, data, indent=2)
```

## jsonschemaparser Module

The JSON schema parser validates a json data structure and if possible casts data to the correct type and fills out default values. The result in a valid document that can be used to construct objects.

```
class fastr.helpers.jsonschemaparser.FastrRefResolver(base_uri, referrer, store=(),
                                                       cache_remote=True, handlers=())
```

Bases: RefResolver

Adapted version of the RefResolver for handling inter-file references more to our liking

```
__init__(base_uri, referrer, store=(), cache_remote=True, handlers=())
```

Create a new FastrRefResolver

### Parameters

- **base\_uri** (*str*) – URI of the referring document
- **referrer** – the actual referring document
- **store** (*dict*) – a mapping from URIs to documents to cache
- **cache\_remote** (*bool*) – whether remote refs should be cached after first resolution
- **handlers** (*dict*) – a mapping from URI schemes to functions that should be used to retrieve them

```
__module__ = 'fastr.helpers.jsonschemaparser'
```

```
classmethod from_schema(schema, *args, **kwargs)
```

Instantiate a RefResolver based on a schema

```
static readfastrschema(name)
```

Open a json file based on a fastr:// url that points to a file in the fastr.schemadir

### Parameters

**name** (*str*) – the url of the file to open

### Returns

the resulting json schema data

```
static readfile(filename)
```

Open a json file based on a simple filename

### Parameters

**filename** (*str*) – the path of the file to read

### Returns

the resulting json schema data

`fastr.helpers.jsonschemaparser.any_of_draft4(validator, any_of, instance, schema)`

The oneOf directory needs to be done stepwise, because a validation even if it fails will try to change types / set defaults etc. Therefore we first create a copy of the data per subschema and test if they match. Then for all the schemas that are valid, we perform the validation on the actual data so that only the valid subschemas will effect the data.

#### Parameters

- **validator** – the json schema validator
- **any\_of** (*dict*) – the current oneOf
- **instance** – the current object instance
- **schema** (*dict*) – the current json schema

`fastr.helpers.jsonschemaparser.extend(validator_cls)`

Extend the given jsonschema.Validator with the Seep layer.

`fastr.helpers.jsonschemaparser.getblueprinter(uri, blueprint=None)`

Instantiate the given data using the blueprinter.

#### Parameters

**blueprint** – a blueprint (JSON Schema with Seep properties)

`fastr.helpers.jsonschemaparser.items_prevalidate(validator, items, instance, schema)`

The pre-validation function for items

#### Parameters

- **validator** – the json schema validator
- **items** (*dict*) – the current items
- **instance** – the current object instance
- **schema** (*dict*) – the current json schema

`fastr.helpers.jsonschemaparser.not_draft4(validator, not_schema, instance, schema)`

The not needs to use a temporary copy of the instance, not to change the instance with the invalid schema

#### Parameters

- **validator** – the json schema validator
- **not\_schema** (*dict*) – the current oneOf
- **instance** – the current object instance
- **schema** (*dict*) – the current json schema

`fastr.helpers.jsonschemaparser.one_of_draft4(validator, one_of, instance, schema)`

The one\_of directory needs to be done stepwise, because a validation even if it fails will try to change types / set defaults etc. Therefore we first create a copy of the data per subschema and test if they match. Once we found a proper match, we only validate that branch on the real data so that only the valid piece of schema will effect the data.

#### Parameters

- **validator** – the json schema validator
- **one\_of** (*dict*) – the current one\_of
- **instance** – the current object instance
- **schema** (*dict*) – the current json schema

```
fastr.helpers.jsonschemaparser.pattern_properties_prevalid(validator, pattern_properties, instance,  
                                                             schema)
```

The pre-validation function for patternProperties

#### Parameters

- **validator** – the json schema validator
- **pattern\_properties** (*dict*) – the current patternProperties
- **instance** (*dict*) – the current object instance
- **schema** (*dict*) – the current json schema

```
fastr.helpers.jsonschemaparser.properties_postvalidate(validator, properties, instance, schema)
```

# All arguments must be used because this function is called like this # pylint: disable=unused-argument The post-validation function for properties

#### Parameters

- **validator** – the json schema validator
- **properties** (*dict*) – the current properties
- **instance** – the current object instance
- **schema** (*dict*) – the current json schema

```
fastr.helpers.jsonschemaparser.properties_prevalidate(validator, properties, instance, schema)
```

The pre-validation function for properties

#### Parameters

- **validator** – the json schema validator
- **properties** (*dict*) – the current properties
- **instance** – the current object instance
- **schema** (*dict*) – the current json schema

## lazy\_module Module

This module contains the Manager class for Plugins in the fastr system

```
class fastr.helpers.lazy_module.LazyModule(name, parent, plugin_manager)
```

Bases: module

A module that allows content to be loaded lazily from plugins. It generally is (almost) empty and gets (partially) populated when an attribute cannot be found. This allows lazy loading and plugins depending on other plugins.

```
__getattr__(item)
```

The getattr is called when getattribute does not return a value and is used as a fallback. In this case we try to find the value normally and will trigger the plugin manager if it cannot be found.

#### Parameters

- **item** (*str*) – attribute to retrieve

#### Returns

the requested attribute

```
__init__(name, parent, plugin_manager)
```

```
__module__ = 'fastr.helpers.lazy_module'

__repr__()
    Return repr(self).
```

## lockfile Module

A module implenting a lock that ensures a directory is only being used by a single fastr run.

**class** fastr.helpers.lockfile.DirectoryLock(*directory*)

Bases: `object`

A lock for a directory, it creates a directory to set the locked state and if successful writes the pid in a file inside that directory to claim the lock

```
__annotations__ = {'lock_dir_name': <class 'str'>, 'pid_file_name': <class 'str'>}

__del__()

__dict__ = mappingproxy({'__module__': 'fastr.helpers.lockfile', '__annotations__':
{'lock_dir_name': <class 'str'>, 'pid_file_name': <class 'str'>}, '__doc__': '\n
A lock for a directory, it creates a directory to set the locked state and\n
if successful writes the pid in a file inside that directory to claim the\n
lock\n
', 'lock_dir_name': '.fastr.lock', 'pid_file_name': 'pid', '__init__': <function
DirectoryLock.__init__>, 'lock_dir': <property object>, 'pid_file': <property
object>, 'get_pid': <function DirectoryLock.get_pid>, '_checkpid': <staticmethod
object>, 'acquire': <function DirectoryLock.acquire>, 'release': <function
DirectoryLock.release>, '__enter__': <function DirectoryLock.__enter__>,
'__exit__': <function DirectoryLock.__exit__>, '__del__': <function
DirectoryLock.__del__>, '__dict__': <attribute '__dict__' of 'DirectoryLock'
objects>, '__weakref__': <attribute '__weakref__' of 'DirectoryLock' objects>}}

__enter__()

__exit__(type, value, traceback)

__init__(directory)

__module__ = 'fastr.helpers.lockfile'

__weakref__
    list of weak references to the object (if defined)

acquire()

    Return type
    bool

get_pid()

    Return type
    Optional[int]

property lock_dir: Path

    Return type
    Path
```

```
lock_dir_name: str = '.fastr.lock'

property pid_file: Path

    Return type
    Path

pid_file_name: str = 'pid'

release(force=False)
```

## procutils Module

`fastr.helpers.procutils.which(name)`

Find executable by name on the PATH, returns the executable that will be found in case it is used for a Popen call

## report Module

Some reporting functions, e.g. to print a report based on a job result

`fastr.helpers.report.print_job_result(job_file, print_func=<built-in function print>, verbose=False)`

## rest\_generation Module

`fastr.helpers.rest_generation.create_rest_table(data, headers)`

Create a ReST table from data. The data should be a list of columns and the headers should be a list of column names.

### Parameters

- **data** (*list*) – List of lists/tuples representing the columns
- **headers** (*list*) – List of strings for the column names

### Returns

a string representing the table in ReST

### Return type

str

## schematortable Module

A module to generate reStructuredText tables from json schema files

`class fastr.helpers.schematortable.SchemaPrinter(schema, skipfirst=False)`

Bases: `object`

Object that create a table in reStructuredText from a json schema



```
__dict__ = mappingproxy({'__module__': 'fastr.helpers.schematatable', '__doc__':
'\n Object that create a table in reStructuredText from a json schema\n ',
 '__init__': <function SchemaPrinter.__init__>, '__str__': <function
SchemaPrinter.__str__>, 'descend': <function SchemaPrinter.descend>, 'parse':
<function SchemaPrinter.parse>, 'printlines': <function SchemaPrinter.printlines>,
 '__dict__': <attribute '__dict__' of 'SchemaPrinter' objects>, '__weakref__':
<attribute '__weakref__' of 'SchemaPrinter' objects>, '__annotations__': {}})
```

**\_\_init\_\_**(*schema*, *skipfirst=False*)

Create the printer object

#### Parameters

- **schema** (*dict*) – the json schema to print
- **skipfirst** (*bool*) – flag to indicate that the first line should not be printed

**\_\_module\_\_** = 'fastr.helpers.schematatable'

**\_\_str\_\_**()

String representation of json schema (that is the printed table)

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**descend**(*properties*)

Descend into a subschema

#### Parameters

**properties** (*dict*) – the properties in the subschema

**parse**(*schema=None*)

Parse a schema

#### Parameters

**schema** (*dict*) – the schema to parse

**printlines**()

Given a parsed schema (parsing happens when the object is constructed), print all the lines

#### Returns

the printed table

#### Return type

*str*

## shellescape Module

Module with helper for shell escaping

**fastr.helpers.shellescape.quote\_argument**(*arg*)

Use shlex module to quote the argument properly :type *arg*: *str* :param *arg*: argument to quote :rtype: *str*  
:return: argument with quotes for safe use in a bash-like shell

## **sysinfo Module**

This module contains function to help gather system information use for the provenance of the Job execution.

`fastr.helpers.sysinfo.get_cpu_usage()`

Get the current CPU usage

**Returns**

CPU usage info

**Return type**

`dict`

`fastr.helpers.sysinfo.get_drmaa_info()`

Get information about the SGE cluster (if applicable)

**Returns**

cluster info

**Return type**

`dict`

`fastr.helpers.sysinfo.get_hostinfo()`

Get all information about the current host machine

**Returns**

host info

**Return type**

`dict`

`fastr.helpers.sysinfo.get_memory_usage()`

Get the current memory usage

**Returns**

memory usage info

**Return type**

`dict`

`fastr.helpers.sysinfo.get_mounts()`

Get the current mounts known on the system

**Returns**

mount info

**Return type**

`dict`

`fastr.helpers.sysinfo.get_os()`

Get information about the OS

**Returns**

OS information

**Return type**

`dict`

`fastr.helpers.sysinfo.get_processes()`

Get a list of all currently running processes

**Returns**

process information

**Return type**

`list`

`fastr.helpers.sysinfo.get_python()`

Get information about the currently used Python implementation

**Returns**

python info

**Return type**

`dict`

`fastr.helpers.sysinfo.get_sysinfo()`

Get system information (cpu, memory, mounts and users)

**Returns**

system information

**Return type**

`dict`

`fastr.helpers.sysinfo.get_users()`

Get current users on the system

**Returns**

user info

**Return type**

`dict`

`fastr.helpers.sysinfo.namedtuple_to_dict(ntuple)`

Helper function to convert a named tuple into a dict

**Parameters**

**ntuple** (*namedtuple*) – the namedtuple to convert

**Returns**

named tuple as a dict

**Return type**

`dict`

## **xmltodict Module**

This module contains tool for converting python dictionaries into XML object and vice-versa.

`fastr.helpers.xmltodict.dump(data, filehandle)`

Write a dict to an XML file

**Parameters**

- **data** – data to write
- **filehandle** – file handle to write to

`fastr.helpers.xmltodict.dumps(data)`

Write a dict to an XML string

**Parameters**

**data** – data to write

**Returns**

the XML data

**Return type**

`str`

`fastr.helpers.xmltodict.load(filehandle)`

Load an xml file and parse it to a dict

**Parameters**

**filehandle** – file handle to load

**Returns**

the parsed data

`fastr.helpers.xmltodict.loads(data)`

Load an xml string and parse it to a dict

**Parameters**

**data** (`str`) – the xml data to load

**Returns**

the parsed data

## planning Package

## planning Package

## inputgroup Module

**class** `fastr.planning.inputgroup.InputGroup(*args, **kwargs)`

Bases: `OrderedDict`, `HasDimensions`

A class representing a group of inputs. Input groups allow the

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_getitem\_\_**(*key*)

`x.__getitem__(y) <==> x[y]`

**\_\_init\_\_**(*\*args, \*\*kwargs*)

Create a new InputGroup representation

**Parameters**

- **parent** (`NodeRun`) – the parent node
- **id** (`str`) – the id of the input group

**Raises**

`FastrTypeError` – if parent is not a `NodeRun`

---

**Note:** This is a wrapped version of `fastr.planning.inputgroup.__init__` which triggers an update of the object after being called

---

---

```
__module__ = 'fastr.planning.inputgroup'
```

```
__setitem__(*args, **kwargs)
```

Assign an input to this input group.

**Parameters**

- **key** (*str*) – id of the input
- **value** (*Input*) – the input to assign

**Raises**

*FastrTypeError* – if value of valid type

---

**Note:** This is a wrapped version of `fastr.planning.inputgroup.__setitem__` which triggers an update of the object after being called

---

```
__updatefunc__()
```

Update the InputGroup. Triggers when a change is made to the content of the InputGroup. Automatically recalculates the size, primary Input etc.

```
__updatetriggers__ = ['__init__', '__setitem__', '__delitem__', 'clear', 'pop',  
'popitem', 'setdefault', 'update']
```

**property dimensions**

The dimensions of this InputGroup

**property empty**

Bool indicating that this InputGroup is empty (has no data connected)

```
find_source_index(target_size, target_dimnames, source_size, source_dimnames, target_index)
```

**property fullid**

**property iterinputvalues**

Iterate over the item in this InputGroup

**Returns**

iterator yielding `SampleItems`

**property parent**

The parent node of this InputGroup

**property primary**

The primary Input in this InputGroup. The primary Input is the Input that defines the size of this InputGroup. In case of ties it will be the first in the tool definition.

```
classmethod solve_broadcast(target_size, target_dimnames, source_size, source_dimnames,  
                             target_index, nodegroups=None)
```

## inputgroupcombiner Module

**class** `fastr.planning.inputgroupcombiner.BaseInputGroupCombiner(parent)`

Bases: *HasDimensions*

An object that takes the different input groups and combines them in the correct way.

**\_\_abstractmethods\_\_** = `frozenset({'iter_input_groups', 'merge', 'unmerge'})`

**\_\_init\_\_**(*parent*)

**\_\_iter\_\_**()

**\_\_module\_\_** = `'fastr.planning.inputgroupcombiner'`

### property dimensions

The dimensions has to be implemented by any subclass. It has to provide a tuple of Dimensions.

#### Returns

dimensions

#### Return type

*tuple*

### property fullid

The full id of the InputGroupCombiner

### property input\_groups

**abstract** `iter_input_groups()`

Iterate over all the merged samples :return:

**abstract** `merge(list_of_items)`

Given a list of items for each input group, it returns the combined list of items.

#### Parameters

**list\_of\_items** (*list*) – items to combine

#### Returns

combined list

**merge\_failed\_annotations**(*list\_of\_failed\_annotations*)

**merge\_payloads**(*sample\_payloads*)

**merge\_sample\_data**(*list\_of\_sample\_data*)

**merge\_sample\_id**(*list\_of\_sample\_ids*)

**merge\_sample\_index**(*list\_of\_sample\_indexes*)

**merge\_sample\_jobs**(*list\_of\_sample\_jobs*)

**merge\_sample\_status**(*states*)

**abstract** `unmerge(item)`

Given a item it will recreate the separate items, basically this is the inverse operation of merge. However, this create an OrderedDict so that specific input groups can be easily retrieved. To get a round trip, the values of the OrderedDict should be taken:

```
>>> odict_of_items = combiner.unmerge(item)
>>> item = combiner.merge(odict_of_items.values())
```

**Parameters**

**item** (*list*) – the item to unmerge

**Returns**

items

**Return type**

OrderedDict

**update()**

**class** fastr.planning.inputgroupcombiner.DefaultInputGroupCombiner(*parent*)

Bases: *BaseInputGroupCombiner*

The default input group combiner combines the input group in a cross product version, taking each combinations of samples between the input groups. So if there are two input groups with one with size N and the other with size M x P the result would be N x M x P samples, with all possible combinations of the samples in each input group.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_module\_\_** = 'fastr.planning.inputgroupcombiner'

**iter\_input\_groups()**

Iterate over all the merged samples :return:

**merge(list\_of\_items)**

Given a list of items for each input group, it returns the combined list of items.

**Parameters**

**list\_of\_items** (*list*) – items to combine

**Returns**

combined list

**unmerge(item)**

Given a item it will recreate the seperate items, basically this is the inverse operation of merge. However, this create an OrderedDict so that specific input groups can be easily retrieved. To get a round trip, the values of the OrderedDict should be taken:

```
>>> odict_of_items = combiner.unmerge(item)
>>> item = combiner.merge(odict_of_items.values())
```

**Parameters**

**item** (*list*) – the item to unmerge

**Returns**

items

**Return type**

OrderedDict

```
class fastr.planning.inputgroupcombiner.MergingInputGroupCombiner(input_groups,
                                                                    merge_dimension)
```

Bases: *BaseInputGroupCombiner*

The merging input group combiner takes a similar approach as the default combiner but merges dimensions that are the same. If input group A has  $N(3) \times M(2)$  samples and B has  $M(2) \times P(4)$  it will not result in  $N(3) \times M(2) \times P(4)$ , but merge the dimensions  $M$  leading to  $N(3) \times M(2) \times P(4)$  in resulting size.

```
__abstractmethods__ = frozenset({})
```

```
__init__(input_groups, merge_dimension)
```

```
__module__ = 'fastr.planning.inputgroupcombiner'
```

```
iter_input_groups()
```

Iterate over all the merged samples :return:

```
merge(list_of_items)
```

Given a list of items for each input group, it returns the combined list of items.

**Parameters**

**list\_of\_items** (*list*) – items to combine

**Returns**

combined list

```
unmerge(item)
```

Given a item it will recreate the separate items, basically this is the inverse operation of merge. However, this creates an OrderedDict so that specific input groups can be easily retrieved. To get a round trip, the values of the OrderedDict should be taken:

```
>>> odict_of_items = combiner.unmerge(item)
>>> item = combiner.merge(odict_of_items.values())
```

**Parameters**

**item** (*list*) – the item to unmerge

**Returns**

items

**Return type**

OrderedDict

```
update()
```

## inputoutput Module

Classes for arranging the input and output for nodes.

Exported classes:

Input – An input for a node (holding datatype). Output – The output of a node (holding datatype and value). ConstantOutput – The output of a node (holding datatype and value).

**Warning:** Don't mess with the Link, Input and Output internals from other places. There will be a huge chance of breaking the network functionality!



**class** `fastr.planning.inputoutput.AdvancedFlowOutput(node, description)`

Bases: `Output`

Output for nodes that have an advanced flow. This means that the output sample id and index is not the same as the input sample id and index. The AdvancedFlowOutput has one extra dimensions that is created by the Node.

`__abstractmethods__ = frozenset({})`

`__module__ = 'fastr.planning.inputoutput'`

**property dimensions**

The list of the dimensions in this Output. This will be a tuple of Dimension.

**class** `fastr.planning.inputoutput.BaseInput(node, description)`

Bases: `BaseInputOutput`

Base class for all inputs.

`__abstractmethods__ = frozenset({'_update', 'dimensions', 'fullid', 'itersubinputs'})`

`__init__(node, description)`

Instantiate a BaseInput

**Parameters**

- **node** – the parent node the input/output belongs to.
- **description** – the ParameterDescription describing the input/output.

**Returns**

the created BaseInput

**Raises**

- `FastrTypeError` – if description is not of class ParameterDescription
- `FastrDataTypeNotAvailableError` – if the DataType requested cannot be found in the types

`__lshift__(other)`

`__module__ = 'fastr.planning.inputoutput'`

`__rrshift__(other)`

`check_cardinality(key=None, planning=False)`

Check if the actual cardinality matches the cardinality specified in the ParameterDescription. Optionally you can use a key to test for a specific sample.

**Parameters**

**key** – sample\_index (tuple of int) or SampleId for desired sample

**Returns**

flag indicating that the cardinality is correct

**Return type**

`bool`

**Raises**

`FastrCardinalityError` – if the Input/Output has an incorrect cardinality description.

**constant\_id()**

The id that should be used for a constant created to serve this input.

**Return type**

`str`

**create\_link\_from(value)****property default**

Default value

**description\_type**

alias of `InputSpec`

**property item\_index****abstract itersubinputs()**

Iterator over the SubInputs

**Returns**

iterator

example:

```
>>> for subinput in input_a.itersubinputs():
    print subinput
```

**class** `fastr.planning.inputoutput.BaseInputOutput`(*node, description*)

Bases: `HasDimensions`, `Updateable`, `Serializable`

Base class for Input and Output classes. It mainly implements the properties to access the data from the underlying `ParameterDescription`.

**\_\_abstractmethods\_\_** = `frozenset({'_update', 'dimensions', 'fullid'})`

**\_\_getstate\_\_()**

Retrieve the state of the `BaseInputOutput`

**Returns**

the state of the object

**Rtype dict****\_\_init\_\_**(*node, description*)

Instantiate a `BaseInputOutput`

**Parameters**

- **node** – the parent node the input/output belongs to.
- **description** – the `ParameterDescription` describing the input/output.

**Returns**

created `BaseInputOutput`

**Raises**

- `FastrTypeError` – if description is not of class `ParameterDescription`
- `FastrDataTypeNotAvailableError` – if the `DataType` requested cannot be found in the types

**`__iter__()`**

This function is blocked to avoid support for iteration using a legacy `__getitem__` method.

**Returns**

None

**Raises**

`FastrNotImplementedError` – always

**`__module__`** = 'fastr.planning.inputoutput'

**`__ne__(other)`**

Check two Node instances for inequality. This is the inverse of `__eq__`

**Parameters**

**other** (`BaseInputOutput`) – the other instances to compare to

**Returns**

True if unequal, False otherwise

**`__repr__()`**

Get a string representation for the Input/Output

**Returns**

the string representation

**Return type**

`str`

**`__setstate__(state)`**

Set the state of the BaseInputOutput by the given state.

**Parameters**

**state** (`dict`) – The state to populate the object with

**Returns**

None

**`cardinality(key=None, job_data=None)`**

Determine the cardinality of this Input/Output. Optionally a key can be given to determine for a sample.

**Parameters**

**key** – key for a specific sample

**Returns**

the cardinality

**Return type**

`int`, `sympy.Symbol`, or `None`

**`check_cardinality(key=None, planning=False)`**

Check if the actual cardinality matches the cardinality specified in the ParameterDescription. Optionally you can use a key to test for a specific sample.

**Parameters**

**key** – `sample_index` (tuple of `int`) or `SampleId` for desired sample

**Returns**

flag indicating that the cardinality is correct

**Return type**

`bool`

**Raises**

***FastrCardinalityError*** – if the Input/Output has an incorrect cardinality description.

**property datatype**

The datatype of this Input/Output

**property description**

The description object of this input/output

**description\_type = None****abstract property fullid**

The fullid of the Input/Output, the fullid should be ununique and makes the object retrievable by the network.

**property id**

Id of the Input/Output

**property node**

The NodeRun to which this Input/Output belongs

**property required**

Flag indicating that the Input/Output is required

**class** `fastr.planning.inputoutput.BaseOutput(node, description)`

Bases: *BaseInputOutput*

Base class for all outputs.

**\_\_abstractmethods\_\_** = `frozenset({'_update', 'dimensions', 'fullid'})`

**\_\_init\_\_**(*node, description*)

Instantiate a BaseOutput

**Parameters**

- **node** – the parent node the output belongs to.
- **description** – the ParameterDescription describing the output.

**Returns**

created BaseOutput

**Raises**

- ***FastrTypeError*** – if description is not of class ParameterDescription
- ***FastrDataTypeNotAvailableError*** – if the DataType requested cannot be found in the types

**\_\_module\_\_** = `'fastr.planning.inputoutput'`

**property automatic**

Flag indicating that the Output is generated automatically without being specified on the command line

**property blocking**

Flag indicating that this Output will cause blocking in the execution

**description\_type**

alias of *OutputSpec*

**class** `fastr.planning.inputoutput.Input`(*node*, *description*)

Bases: `BaseInput`

Class representing an input of a node. Such an input will be connected to the output of another node or the output of an constant node to provide the input value.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_eq\_\_**(*other*)

Compare two Input instances with each other. This function ignores the parent node and update status, but tests rest of the dict for equality.

**Parameters**

**other** (`Input`) – the other instances to compare to

**Returns**

True if equal, False otherwise

**Return type**

`bool`

**\_\_getitem\_\_**(*key*)

Retrieve an item from this Input.

**Parameters**

**key** (`Union[int, str]`) – the key of the requested item

**Return type**

`Union[SubInput, NamedSubInput]`

**Returns**

The `SubInput` corresponding with the key will be returned.

**Raises**

- `FastrTypeError` – if key is not of a valid type
- `FastrKeyError` – if the key is not found

**\_\_getstate\_\_**()

Retrieve the state of the Input

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(*node*, *description*)

Instantiate an input.

**Parameters**

- **node** (`NodeRun`) – the parent node of this input.
- **description** (`ParameterDescription`) – the `ParameterDescription` of the input.

**Returns**

the created Input

**\_\_module\_\_** = `'fastr.planning.inputoutput'`

**\_\_setitem\_\_**(*key, value*)

Create a link between a SubInput of this Inputs and an Output/Constant

**Parameters**

- **key** (*int*, *str*) – the key of the SubInput
- **value** (*BaseOutput*, *list*, *tuple*, *dict*, *OrderedDict*) – the target to link, can be an output or a value to create a constant for

**Raises**

*FastrTypeError* – if key is not of a valid type

**\_\_setstate\_\_**(*state*)

Set the state of the Input by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**\_\_str\_\_**()

Get a string version for the Input

**Returns**

the string version

**Return type**

*str*

**append**(*value*)

When you want to append a link to an Input, you can use the append property. This will automatically create a new SubInput to link to.

example:

```
>>> link = node2['input'].append(node1['output'])
```

will create a new SubInput in node2['input'] and link to that.

**cardinality**(*key=None, job\_data=None*)

Cardinality for an Input is the sum the cardinalities of the SubInputs, unless defined otherwise.

**Parameters**

**key** (tuple of *int* or *SampleId*) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

*int*, *sympy.Symbol*, or *None*

**clear**()

**property constant\_id:** *str*

The id for a constant node that is attached to this input.

**Return type**

*str*

**property datatype**

The datatype of this Input

**property dimensions**

The list names of the dimensions in this Input. This will be a list of str.

**property fullid: str**

The full defining ID for the Input

**Return type**

str

**get\_sourced\_nodes()**

Get a list of all *Nodes* connected as sources to this Input

**Returns**

list of all connected *Nodes*

**Return type**

list

**get\_sourced\_outputs()**

Get a list of all *Outputs* connected as sources to this Input

**Returns**

tuple of all connected *Outputs*

**Return type**

tuple

**property id**

Id of the Input/Output

**index(value)**

Find index of a SubInput

**Parameters**

**value** (*SubInput*) – the *SubInput* to find the index of

**Returns**

key

**Return type**

int, str

**property input\_group: str**

The id of the InputGroup this Input belongs to.

**Return type**

str

**insert(index)**

Insert a new SubInput at index in the sources list

**Parameters**

**key** (*int*) – positive integer for position in \_source list to insert to

**Returns**

newly inserted *SubInput*

**Return type**

*SubInput*

**itersubinputs()**

Iterate over the *SubInputs* in this Input.

**Returns**

iterator yielding *SubInput*

example:

```
>>> for subinput in input_a.itersubinputs():
    print subinput
```

**remove(value)**

Remove a SubInput from the SubInputs list based on the connected Link.

**Parameters**

**value** (*SubInput*, <fastr.planning.inputoutput.SubInput>) – the *SubInput* or *Link* to removed from this Input

**property source**

The mapping of *SubInputs* that are connected and have more than 0 elements.

**class** fastr.planning.inputoutput.**MacroInput**(*node, description*)

Bases: *Input*

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_module\_\_** = 'fastr.planning.inputoutput'

**property input\_group**

The id of the InputGroup this Input belongs to.

**class** fastr.planning.inputoutput.**MacroOutput**(*node, description*)

Bases: *Output*

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_module\_\_** = 'fastr.planning.inputoutput'

**property dimensions**

The list of the dimensions in this Output. This will be a tuple of Dimension.

**class** fastr.planning.inputoutput.**NamedSubInput**(*parent*)

Bases: *Input*

A named subinput for cases where the value of an input is mapping.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_getitem\_\_**(*key*)

Retrieve an item (a SubInput) from this NamedSubInput.

**Parameters**

**key** (*int*) – the key of the requested item

**Return type**

*SubInput*

**Returns**

The *SubInput* corresponding with the key will be returned.

**Raises**



- *FastrTypeError* – if key is not of a valid type
- *FastrKeyError* – if the key is not found

**\_\_init\_\_**(*parent*)

Instantiate an input.

**Parameters**

- **node** (*NodeRun*) – the parent node of this input.
- **description** (*ParameterDescription*) – the *ParameterDescription* of the input.

**Returns**

the created Input

**\_\_module\_\_** = 'fastr.planning.inputoutput'

**\_\_str\_\_**()

Get a string version for the *NamedSubInput*

**Returns**

the string version

**Return type**

str

**property constant\_id:** str

The id for a constant node that is attached to this input.

**Return type**

str

**property fullid**

The full defining ID for the *SubInput*

**property item\_index**

**class** fastr.planning.inputoutput.**Output**(*node, description*)

Bases: *BaseOutput*

Class representing an output of a node. It holds the output values of the tool ran. Output fields can be connected to inputs of other nodes.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_eq\_\_**(*other*)

Compare two *Output* instances with each other. This function ignores the parent node, listeners and update status, but tests rest of the dict for equality.

**Parameters**

- **other** (fastr.planning.inputoutput.*Output*) – the other instances to compare to

**Returns**

True if equal, False otherwise

**Return type**

bool

**\_\_getitem\_\_**(*key*)

Retrieve an item from this *Output*. The returned value depends on what type of key used:

- Retrieving data using index tuple: [index\_tuple]

- Retrieving data sample\_id str: [SampleId]
- Retrieving a list of data using SampleId list: [sample\_id1, ..., sample\_idN]
- Retrieving a *SubOutput* using an int or slice: [n] or [n:m]

**Parameters**

**key** (`Union[int, slice]`) – the key of the requested suboutput, can be a number or slice

**Return type**

*SubOutput*

**Returns**

the *SubOutput* for the corresponding index

**Raises**

*FastrTypeError* – if key is not of a valid type

`__getstate__()`

Retrieve the state of the Output

**Returns**

the state of the object

**Rtype dict**

`__hash__ = None`

`__init__(node, description)`

Instantiate an Output

**Parameters**

- **node** – the parent node the output belongs to.
- **description** – the ParameterDescription describing the output.

**Returns**

created Output

**Raises**

- *FastrTypeError* – if description is not of class ParameterDescription
- *FastrDataTypeNotAvailableError* – if the DataType requested cannot be found in the types

`__module__ = 'fastr.planning.inputoutput'`

`__setstate__(state)`

Set the state of the Output by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

`__str__()`

Get a string version for the Output

**Returns**

the string version

**Return type**

str

**cardinality()**

Cardinality of this Output, may depend on the inputs of the parent Node.

**Returns**

the cardinality

**Return type**

int, sympy.Symbol, or None

**Raises**

- **FastrCardinalityError** – if cardinality references an invalid *Input*
- **FastrTypeError** – if the referenced cardinality values type cannot be case to int
- **FastrValueError** – if the referenced cardinality value cannot be case to int

**property datatype**

The datatype of this Output

**property dimensions**

The list of the dimensions in this Output. This will be a tuple of Dimension.

**property fullid**

The full defining ID for the Output

**property listeners**

The list of *Links* connected to this Output.

**property preferred\_types**

The list of preferred DataTypes for this Output.

**property resulting\_datatype**

The DataType that will the results of this Output will have.

**property valid**

Check if the output is valid, i.e. has a valid cardinality

**class** fastr.planning.inputoutput.SourceOutput(*node, description*)

Bases: *Output*

Output for a SourceNodeRun, this type of Output determines the cardinality in a different way than a normal NodeRun.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_init\_\_**(*node, description*)

Instantiate a FlowOutput

**Parameters**

- **node** – the parent node the output belongs to.
- **description** – the ParameterDescription describing the output.

**Returns**

created FlowOutput

**Raises**

- *FastrTypeError* – if description is not of class *ParameterDescription*
- *FastrDataTypeNotAvailableError* – if the *DataType* requested cannot be found in the types

`__module__ = 'fastr.planning.inputoutput'`

**cardinality()**

Cardinality of this *SourceOutput*, may depend on the inputs of the parent *NodeRun*.

**Parameters**

**key** (tuple of int or *SampleId*) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

int, *sympy.Symbol*, or None

**property linearized**

A linearized version of the sample data, this is lazily cached linearized version of the underlying *SampleCollection*.

**class** *fastr.planning.inputoutput.SubInput*(*input\_*)

Bases: *BaseInput*

This class is used by *Input* to allow for multiple links to an *Input*. The *SubInput* class can hold only a single Link to a (Sub)Output, but behaves very similar to an *Input* otherwise.

`__abstractmethods__ = frozenset({})`

`__eq__(other)`

Compare two *SubInput* instances with each other. This function ignores the parent, node, source and update status, but tests rest of the dict for equality.

**Parameters**

**other** (*SubInput*) – the other instances to compare to

**Returns**

True if equal, False otherwise

`__getitem__(key)`

Retrieve an item from this *SubInput*.

**Parameters**

**key** (*int*) – the index of the requested item

**Returns**

the corresponding *SubInput*

**Return type**

*SubInput*

**Raises**

*FastrTypeError* – if key is not of a valid type

---

**Note:** As a *SubInput* has only one *SubInput*, only requesting int key 0 or -1 is allowed, and it will return self

---

**\_\_getstate\_\_()**

Retrieve the state of the SubInput

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_ = None**

**\_\_init\_\_(input\_)**

Instantiate an SubInput.

**Parameters**

**input** (*Input*) – the parent of this SubInput.

**Returns**

the created SubInput

**\_\_module\_\_ = 'fastr.planning.inputoutput'**

**\_\_setstate\_\_(state)**

Set the state of the SubInput by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**\_\_str\_\_()**

Get a string version for the SubInput

**Returns**

the string version

**Return type**

*str*

**cardinality(key=None, job\_data=None)**

Get the cardinality for this SubInput. The cardinality for a SubInputs is defined by the incoming link.

**Parameters**

**key** (*SampleIndex* or *SampleId*) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

*int*, *sympy.Symbol*, or *None*

**property constant\_id: str**

The id for a constant node that is attached to this input.

**Return type**

*str*

**property description**

The description object of this input/output

**property dimensions**

List of dimension for this SubInput

**property fullid**

The full defining ID for the SubInput

**get\_sourced\_nodes()**

Get a list of all *Nodes* connected as sources to this SubInput

**Returns**

list of all connected *Nodes*

**Return type**

list

**get\_sourced\_outputs()**

Get a list of all *Outputs* connected as sources to this SubInput

**Returns**

list of all connected *Outputs*

**Return type**

list

**property input\_group**

The id of the InputGroup this SubInputs parent belongs to.

**property item\_index****iteritems()**

Iterate over the SampleItems that are in the SubInput.

**Returns**

iterator yielding SampleItem objects

**itersubinputs()**

Iterate over SubInputs (for a SubInput it will yield self and stop iterating after that)

**Returns**

iterator yielding *SubInput*

example:

```
>>> for subinput in input_a.itersubinputs():
    print subinput
```

**property node**

The Node to which this SubInputs parent belongs

**remove(value)**

Remove a SubInput from parent Input.

**Parameters**

**value** (*SubInput*) – the *SubInput* to removed from this Input

**property source**

A list with the source *Link*. The list is to be compatible with *Input*

**property source\_output**

The *Output* linked to this SubInput

**class** `fastr.planning.inputoutput.SubOutput(output, index)`

Bases: `Output`

The SubOutput is an Output that represents a slice of another Output.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_eq\_\_**(other)

Compare two SubOutput instances with each other. This function ignores the parent, node and update status, but tests rest of the dict for equality. equality

**Parameters**

**other** (`SubOutput`) – the other instances to compare to

**Returns**

True if equal, False otherwise

**Return type**

`bool`

**\_\_getstate\_\_**()

Retrieve the state of the SubOutput

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(output, index)

Instantiate a SubOutput

**Parameters**

- **output** – the parent output the suboutput slices.
- **index** (`int` or `slice`) – the way to slice the parent output

**Returns**

created SubOutput

**Raises**

- `FastrTypeError` – if the output argument is not an instance of `Output`
- `FastrTypeError` – if the index argument is not an `int` or `slice`

**\_\_len\_\_**()

Return the length of the Output.

---

**Note:** In a SubOutput this is always 1.

---

**\_\_module\_\_** = `'fastr.planning.inputoutput'`

**\_\_setstate\_\_**(state)

Set the state of the SubOutput by the given state.

**Parameters**

**state** (`dict`) – The state to populate the object with

**Returns**

None

**\_\_str\_\_()**

Get a string version for the SubOutput

**Returns**

the string version

**Return type**

str

**cardinality()**

Cardinality of this SubOutput depends on the parent Output and `self.index`

**Parameters**

**key** (tuple of int or `SampleId`) – key for a specific sample, can be sample index or id

**Returns**

the cardinality

**Return type**

int, sympy.Symbol, or None

**Raises**

- ***FastrCardinalityError*** – if cardinality references an invalid *Input*
- ***FastrTypeError*** – if the referenced cardinality values type cannot be case to int
- ***FastrValueError*** – if the referenced cardinality value cannot be case to int

**property datatype**

The datatype of this SubOutput

**property fullid**

The full defining ID for the SubOutput

**property indexrep**

Simple representation of the index.

**property listeners**

The list of *Links* connected to this Output.

**property node**

The NodeRun to which this SubOutput belongs

**property preferred\_types**

The list of preferred DataTypes for this SubOutput.

**property resulting\_datatype**

The DataType that will the results of this SubOutput will have.

**property samples**

The SampleCollection for this SubOutput



## link Module

The link module contain the Link class. This class represents the links in a network. These links lead from an output (BaseOutput) to an input (BaseInput) and indicate the desired data flow. Links are smart objects, in the sense that when you set their start or end point, they register themselves with the Input and Output. They do all the book keeping, so as long as you only set the source and target of the Link, the link should be valid.

**Warning:** Don't mess with the Link, Input and Output internals from other places. There will be a huge chances of breaking the network functionality!

**class** fastr.planning.link.Link(*source, target, parent, id\_=None, collapse=None, expand=None*)

Bases: *HasDimensions*, Updateable, Serializable

Class for linking outputs (*BaseOutput*) to inputs (*BaseInput*)

Examples:

```
>>> import fastr
>>> network = fastr.create_network()
>>> link1 = network.create_link( n1.ouputs['out1'], n2.inputs['in2'] )

link2 = Link()
link2.source = n1.ouputs['out1']
link2.target = n2.inputs['in2']
```

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_dataschemafile\_\_** = 'Link.schema.json'

**\_\_eq\_\_**(*other*)

Test for equality between two Links

**Parameters**

**other** (*Link*) – object to test against

**Return type**

bool

**Returns**

True for equality, False otherwise

**\_\_getstate\_\_**()

Retrieve the state of the Link

**Return type**

dict

**Returns**

the state of the object

**\_\_hash\_\_** = None

**\_\_init\_\_**(*source, target, parent, id\_=None, collapse=None, expand=None*)

Create a new Link in a Network.

**Parameters**

- **source** (*BaseOutput*) – the source output

- **target** (*BaseInput*) – the target input
- **parent** (*Network* or None) – the parent network
- **id** – the id of the link, if no **id\_** is given, the id will be in the form of “link\_{:d}”
- **collapse** (*Union[str, int, Tuple[str, ...], None]*) – the dimensions that the link has to collapse on
- **expand** (*Optional[bool]*) – Does this link need to expand the cardinality into a new sample dimension

**Returns**

newly created Link

**Raises**

- *FastrValueError* – if parent is not given
- *FastrValueError* – if the source output is not in the same network as the Link
- *FastrValueError* – if the target input is not in the same network as the Link

**\_\_module\_\_** = 'fastr.planning.link'

**\_\_repr\_\_**()

Get a string representation for the Link

**Return type**

*str*

**\_\_setstate\_\_**(*state*)

Set the state of the Link by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**Raises**

*FastrValueError* – if the parent network is not set

**cardinality**(*index=None*)

Cardinality for a Link is given by source Output and the collapse/expand settings

**Parameters**

**index** (*Optional[SampleIndex]*) – index for a specific sample (can be only a sample index!)

**Return type**

*Union[int, Symbol]*

**Returns**

the cardinality

**Raises**

*FastrIndexError* – if the index length does not match the number of dimension in the data

**property collapse**

The converging dimensions of this link. Collapsing changes some dimensions of sample lists into cardinality, reshaping the data.

Collapse can be set to a tuple or an int/str, in which case it will be automatically wrapped in a tuple. The int will be seen as indices of the dimensions to collapse. The str will be seen as the name of the dimensions over which to collapse.

**Raises**

*FastrTypeError* – if assigning a collapse value of a wrong type

**property collapse\_indexes**

The converging dimensions of this link as integers. Dimension names are replaced with the corresponding int.

Collapsing changes some dimensions of sample lists into cardinality, reshaping the data

**classmethod createobj(*state, network=None*)**

Create object function for Link

**Parameters**

- **cls** – The class to create
- **state** (*dict*) – The state to use to create the Link
- **network** – the parent Network

**Returns**

newly created Link

**destroy()**

The destroy function of a link removes all default references to a link. This means the references in the network, input and output connected to this link. If there is no references in other places in the code, it will destroy the link (reference count dropping to zero).

This function is called when a source for an input is set to another value and the links becomes disconnected. This makes sure there is no dangling links.

**property dimensions**

The dimensions of the data delivered by the link. This can be different from the source dimensions because the link can make data collapse or expand.

**draw(*context, graph*)**

**property expand**

Flag indicating that the link will expand the cardinality into a new sample dimension to be created.

**property fullid**

The full defining ID for the Input

**property parent**

The Network to which this Link belongs.

**property source**

The source *BaseOutput* of the Link. Setting the source will automatically register the Link with the source BaseOutput. Updating source will also make sure the Link is unregistered with the previous source.

**Raises**

*FastrTypeError* – if assigning a non *BaseOutput*

**property status**

**property target**

The target *BaseInput* of the Link. Setting the target will automatically register the Link with the target BaseInput. Updating target will also make sure the Link is unregistered with the previous target.

**Raises**

*FastrTypeError* – if assigning a non *BaseInput*

**network Module**

Network module containing Network facilitators and analysers.

**class** fastr.planning.network.**Network**(*id\_='unnamed\_network', version=None, filename=None*)

Bases: Serializable

The NetworkRun contains the entire Run state for a Network execution. It has a working copy of the network, but also includes all temporary data required for the execution. These objects are meant to be single use.

**NETWORK\_DUMP\_FILE\_NAME** = '\_\_fastr\_network\_\_.yaml'

**SINK\_DUMP\_FILE\_NAME** = '\_\_sink\_data\_\_.json'

**SOURCE\_DUMP\_FILE\_NAME** = '\_\_source\_data\_\_.pickle.gz'

**\_\_dataschemafile\_\_** = 'Network.schema.json'

**\_\_eq\_\_**(*other*)

Compare two Networks and see if they are equal.

**Parameters**

**other** (*Network*) –

**Returns**

flag indicating that the Networks are the same

**Return type**

bool

**\_\_getitem\_\_**(*item*)

Get an item by its fullid. The fullid can point to a link, node, input, output or even subinput/suboutput.

**Parameters**

**item** (*str, unicode*) – fullid of the item to retrieve

**Returns**

the requested item

**\_\_getstate\_\_**()

Retrieve the state of the Network

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = None

**\_\_init\_\_**(*id\_='unnamed\_network', version=None, filename=None*)

Create a new, empty Network

**Parameters**

**name** (*str*) – name of the Network

**Returns**

newly created Network

**Raises**

**OSError** – if the tmp mount in the config is not a writable directory

**\_\_module\_\_** = 'fastr.planning.network'

**\_\_ne\_\_**(*other*)

Tests for non-equality, this is the negated version **\_\_eq\_\_**

**\_\_repr\_\_**()

Return repr(self).

**\_\_setstate\_\_**(*state*)

Set the state of the Network by the given state. This completely overwrites the old state!

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**add\_link**(*link*)

Add a Link to the Network. Make sure the link is in the link list and the link parent is set to this Network

**Parameters**

**link** (*Link*) – link to add

**Raises**

- **FastrTypeError** – if link is incorrectly typed
- **FastrNetworkMismatchError** – if the link already belongs to another Network

**add\_node**(*node*)

Add a Node to the Network. Make sure the node is in the node list and the node parent is set to this Network

**Parameters**

**node** (*Node*) – node to add

**Raises**

**FastrTypeError** – if node is incorrectly typed

**add\_stepid**(*stepid*, *node*)

Add a Node to a specific step id

**Parameters**

- **stepid** (*str*) – the stepid that the node will be added to
- **node** (*Node*) – the node to add to the stepid

**check\_id**(*id\_*)

Check if an id for an object is valid and unused in the Network. The method will always returns True if it does not raise an exception.

**Parameters**

**id** (*str*) – the id to check

**Returns**

True

**Raises**

- **FastrValueError** – if the id is not correctly formatted
- **FastrValueError** – if the id is already in use

**create\_constant**(*datatype*, *data*, *id\_=None*, *stepid=None*, *resources=None*, *nodegroup=None*)

Create a ConstantNode in this Network. The Node will be automatically added to the Network.

#### Parameters

- **datatype** (*BaseDataType*) – The DataType of the constant node
- **data** (*datatype or list of datatype*) – The data to hold in the constant node
- **id** (*str*) – The id of the constant node to be created
- **stepid** (*str*) – The stepid to add the created constant node to
- **resources** – The resources required to run this node
- **nodegroup** (*str*) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.

#### Returns

the newly created constant node

#### Return type

*ConstantNode*

**create\_link**(*source*, *target*, *id\_=None*, *collapse=None*, *expand=None*)

Create a link between two Nodes and add it to the current Network.

#### Parameters

- **source** (*BaseOutput*) – the output that is the source of the link
- **target** (*BaseInput*) – the input that is the target of the link
- **id** (*str*) – the id of the link

#### Returns

the created link

#### Type

*Link*

**create\_macro**(*network*, *resources=None*, *id\_=None*)

**create\_node**(*tool*, *tool\_version*, *id\_=None*, *stepid=None*, *resources=None*, *nodegroup=None*)

Create a Node in this Network. The Node will be automatically added to the Network.

#### Parameters

- **tool** (*Tool*) – The Tool to base the Node on
- **id** (*str*) – The id of the node to be created
- **stepid** (*str*) – The stepid to add the created node to
- **resources** – The resources required to run this node
- **nodegroup** (*str*) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.

#### Returns

the newly created node

**Return type***Node***create\_reference**(*source\_data*, *output\_directory*)**create\_sink**(*datatype*, *id\_=None*, *stepid=None*, *resources=None*, *nodegroup=None*)

Create a SinkNode in this Network. The Node will be automatically added to the Network.

**Parameters**

- **datatype** (BaseDataType) – The DataType of the sink node
- **id** (*str*) – The id of the sink node to be created
- **stepid** (*str*) – The stepid to add the created sink node to
- **resources** – The resources required to run this node

**Returns**

the newly created sink node

**Return type***SinkNode***create\_source**(*datatype*, *id\_=None*, *stepid=None*, *resources=None*, *nodegroup=None*)

Create a SourceNode in this Network. The Node will be automatically added to the Network.

**Parameters**

- **datatype** (BaseDataType) – The DataType of the source source\_node
- **id** (*str*) – The id of the source source\_node to be created
- **stepid** (*str*) – The stepid to add the created source source\_node to
- **resources** – The resources required to run this node
- **nodegroup** (*str*) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.

**Returns**

the newly created source source\_node

**Return type**

SourceNode

**dependencies**()**draw**(*name=None*, *draw\_dimensions=True*, *hide\_unconnected=True*, *context=None*, *graph=None*, *expand\_macro=False*, *font\_size=14*)**draw\_network**(*name='network\_layout'*, *img\_format='svg'*, *draw\_dimension=True*, *hide\_unconnected=True*, *expand\_macro=False*, *font\_size=14*)

Output a dot file and try to convert it to an image file.

**Parameters****img\_format** (*str*) – extension of the image format to convert to**Returns**

path of the image created or None if failed

**Return type***str* or None

**execute**(*sourcedata*, *sinkdata*, *blocking=True*, *\*\*kwargs*)

**property fullid**

The fullid of the Network, within the network scope

**property global\_id**

The global id of the Network, this is different for networks used in macronodes, as they still have parents.

**property id**

The id of the Network. This is a read only property.

**is\_valid()**

**namespace**

The namespace this network lives in, this will be set by the NetworkManager on load

**property nodegroups**

Give an overview of the nodegroups in the network

**property ns\_id**

The namespace and id of the Tool

**remove**(*value*)

Remove an item from the Network.

**Parameters**

**value** (*Node* or *Link*) – the item to remove

**classmethod test**(*reference\_data\_dir*, *network=None*, *source\_data=None*, *force\_remove\_temp=False*, *tmp\_results\_dir=None*)

Execute the network with the source data specified and test the results against the reference data. This effectively tests the network execution.

**Parameters**

- **reference\_data\_dir** (*str*) – The path or vfs url of reference data to compare with
- **source\_data** (*dict*) – The source data to use
- **force\_remove\_temp** – Make sure the tmp results directory is cleaned at end of test
- **tmp\_results\_dir** – Path to results directory

## node Module

A module to maintain a network node.

Exported classes:

Node – A class encapsulating a tool. ConstantNode – A node encapsulating an Output to set scalar values. SourceNode – A class providing a handle to a file.

**class** `fastr.planning.node.AdvancedFlowNode`(*tool*, *id\_=None*, *parent=None*, *resource\_limits=None*, *nodegroup=None*)

Bases: *FlowNode*

`__abstractmethods__` = `frozenset({})`

`__module__` = `'fastr.planning.node'`



```
class fastr.planning.node.BaseNode
```

Bases: *HasDimensions*, Updateable, Serializable

```
NODE_TYPES = {'AdvancedFlowNode': <class 'fastr.planning.node.AdvancedFlowNode'>,
'ConstantNode': <class 'fastr.planning.node.ConstantNode'>, 'FlowNode': <class
'fastr.planning.node.FlowNode'>, 'MacroNode': <class
'fastr.planning.node.MacroNode'>, 'Node': <class 'fastr.planning.node.Node'>,
'SinkNode': <class 'fastr.planning.node.SinkNode'>, 'SourceNode': <class
'fastr.planning.node.SourceNode'>}
```

```
__abstractmethods__ = frozenset({'_update', 'dimensions'})
```

```
classmethod __init_subclass__(**kwargs)
```

Register nodes in class for easy location

```
__module__ = 'fastr.planning.node'
```

```
class fastr.planning.node.ConstantNode(datatype, data, id_=None, parent=None, resource_limits=None,
nodegroup=None)
```

Bases: *SourceNode*

Class encapsulating one output for which a value can be set. For example used to set a scalar value to the input of a node.

```
__abstractmethods__ = frozenset({})
```

```
__dataschemafile__ = 'ConstantNode.schema.json'
```

```
__getstate__()
```

Retrieve the state of the ConstantNode

#### Returns

the state of the object

#### Rtype dict

```
__init__(datatype, data, id_=None, parent=None, resource_limits=None, nodegroup=None)
```

Instantiation of the ConstantNode.

#### Parameters

- **datatype** – The datatype of the output.
- **data** – the prefilled data to use.
- **id** – The url pattern.

This class should never be instantiated directly (unless you know what you are doing). Instead create a constant using the network class like shown in the usage example below.

usage example:

```
>>> import fastr
>>> network = fastr.create_network()
>>> source = network.create_source(datatype=types['ITKImageFile'], id_='sourceN
↪')
```

or alternatively create a constant node by assigning data to an item in an InputDict:

```
>>> node_a.inputs['in'] = ['some', 'data']
```

which automatically creates and links a `ConstantNode` to the specified Input

**`__module__ = 'fastr.planning.node'`**

**`__setstate__(state)`**

Set the state of the `ConstantNode` by the given state.

**Parameters**

**`state`** (*dict*) – The state to populate the object with

**Returns**

None

**property data**

The data stored in this constant node

**`draw(context, graph, color=None)`**

**property print\_value**

**`set_data(data=None, ids=None)`**

Set the data of this constant node in the correct way. This is mainly for compatibility with the parent class `SourceNode`

**Parameters**

- **`data`** (*dict* or *list of urls*) – the data to use
- **`ids`** – if data is a list, a list of accompanying ids

**`class fastr.planning.node.FlowNode(tool, id_=None, parent=None, resource_limits=None, nodegroup=None)`**

Bases: [Node](#)

A Flow Node is a special subclass of Nodes in which the amount of samples can vary per Output. This allows non-default data flows.

**`__abstractmethods__ = frozenset({})`**

**`__init__(tool, id_=None, parent=None, resource_limits=None, nodegroup=None)`**

Instantiate a flow node.

**Parameters**

- **`tool`** (*Tool*) – The tool to base the node on
- **`id`** (*str*) – the id of the node
- **`parent`** (*Network*) – the parent network of the node

**Returns**

the newly created `FlowNode`

**`__module__ = 'fastr.planning.node'`**

**property blocking**

A `FlowNode` is (for the moment) always considered blocking.

**Returns**

True

**property dimensions**

Names of the dimensions in the Node output. These will be reflected in the `SampleIdList` of this Node.

**property outputsize**

Size of the outputs in this Node

**class** `fastr.planning.node.InputDict`

Bases: `OrderedDict`

The container containing the Inputs of Node. Implements helper functions for the easy linking syntax.

**\_\_module\_\_** = `'fastr.planning.node'`

**\_\_setitem\_\_**(*key*, *value*)

Set an item in the input dictionary. The behaviour depends on the type of the value. For a `BaseInput`, the input will simply be added to the list of inputs. For a `BaseOutput`, a link between the output and input will be created.

**Parameters**

- **key** (*str*) – id of the input to assign/link
- **value** (`BaseInput` or `BaseOutput`) – either the input to add or the output to link

**class** `fastr.planning.node.MacroNode`(*value*, *id\_=None*, *parent=None*, *resource\_limits=None*, *nodegroup=None*)

Bases: `Node`

MacroNode encapsulates an entire network in a single node.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_eq\_\_**(*other*)

Compare two MacroNode instances with each other. This function ignores the parent and update status, but tests rest of the dict for equality. equality

**Parameters**

**other** (`MacroNode`) – the other instances to compare to

**Returns**

True if equal, False otherwise

**\_\_getstate\_\_**()

Retrieve the state of the MacroNode

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(*value*, *id\_=None*, *parent=None*, *resource\_limits=None*, *nodegroup=None*)

**Parameters**

**value** – network to create macronode for

**\_\_module\_\_** = `'fastr.planning.node'`

**\_\_setstate\_\_**(*state*)

Set the state of the Node by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**draw**(*context*, *graph*, *color=None*)

**draw\_link\_target**(*context*, *port\_name*, *input=True*)

**get\_output\_info**(*output*)

This functions maps the output dimensions based on the input dimensions of the macro. This is cached for speed as this can become rather costly otherwise

**Parameters**

**output** – output to get info for

**Returns**

tuple of Dimensions

**property network**

**class** `fastr.planning.node.Node`(*tool*, *id\_=None*, *node\_class=None*, *parent=None*, *resource\_limits=None*, *nodegroup=None*)

Bases: [`BaseNode`](#)

The class encapsulating a node in the network. The node is responsible for setting and checking inputs and outputs based on the description provided by a tool instance.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_dataschemafile\_\_** = `'Node.schema.json'`

**\_\_eq\_\_**(*other*)

Check two Node instances for equality.

**Parameters**

**other** ([`fastr.planning.node.Node`](#)) – the other instances to compare to

**Returns**

True if equal, False otherwise

**\_\_getstate\_\_**()

Retrieve the state of the Node

**Returns**

the state of the object

**Rtype dict**

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(*tool*, *id\_=None*, *node\_class=None*, *parent=None*, *resource\_limits=None*, *nodegroup=None*)

Instantiate a node.

**Parameters**

- **tool** ([`Tool`](#)) – The tool to base the node on
- **id** ([`str`](#)) – the id of the node
- **node\_class** ([`str`](#)) – The class of the NodeRun to create (e.g. `SourceNodeRun`, `NodeRun`)
- **parent** ([`Network`](#)) – the parent network of the node

**Returns**

the newly created Node

**\_\_module\_\_** = 'fastr.planning.node'

**\_\_ne\_\_**(*other*)

Check two Node instances for inequality. This is the inverse of **\_\_eq\_\_**

**Parameters**

**other** (*fastr.planning.node.Node*) – the other instances to compare to

**Returns**

True if unequal, False otherwise

**\_\_repr\_\_**()

Get a string representation for the Node

**Returns**

the string representation

**Return type**

*str*

**\_\_setstate\_\_**(*state*)

Set the state of the Node by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**\_\_str\_\_**()

Get a string version for the Node

**Returns**

the string version

**Return type**

*str*

**property blocking**

Indicate that the results of this Node cannot be determined without first executing the Node, causing a blockage in the creation of jobs. A blocking Nodes causes the Chunk borders.

**classmethod createobj**(*state, network=None*)

Create object function for generic objects

**Parameters**

- **cls** – The class to create
- **state** – The state to use to create the Link

**Returns**

newly created Link

**property dimensions**

The dimensions has to be implemented by any subclass. It has to provide a tuple of Dimensions.

**Returns**

dimensions

**Return type**

`tuple`

**property dimnames**

Names of the dimensions in the Node output. These will be reflected in the SampleIdList of this Node.

**draw**(*context*, *graph*, *color=None*)

**draw\_id**(*context*)

**draw\_link\_target**(*context*, *port\_name*, *input=True*)

**find\_source\_index**(*target\_index*, *target*, *source*)

**property fullid**

The full defining ID for the Node inside the network

**get\_sourced\_nodes**()

A list of all Nodes connected as sources to this Node

**Returns**

list of all nodes that are connected to an input of this node

**property global\_id**

The global defining ID for the Node from the main network (goes out of macro nodes to root network)

**property id**

The id of the Node

**property input\_groups**

**A list of input groups for this Node. An input group is InputGroup**  
object filled according to the Node

**inputs**

A list of inputs of this Node

**property listeners**

All the listeners requesting output of this node, this means the listeners of all Outputs and SubOutputs

**property merge\_dimensions**

**property name**

Name of the Tool the Node was based on. In case a Toolless Node was used the class name is given.

**property nodegroup**

**outputs**

A list of outputs of this Node

**property outputsize**

The size of output of this SourceNode

**property parent**

The parent is the Network this Node is part of

**property status**

**property tool**

**update\_input\_groups()**

Update all input groups in this node

**class fastr.planning.node.OutputDict**

Bases: `OrderedDict`

The container containing the Inputs of Node. Only checks if the inserted values are actually outputs.

**\_\_module\_\_** = 'fastr.planning.node'

**\_\_setitem\_\_**(*key*, *value*)

Set an output.

**Parameters**

- **key** (*str*) – the of the item to set
- **value** (*BaseOutput*) – the output to set

**class fastr.planning.node.SinkNode**(*datatype*, *id\_=None*, *parent=None*, *resource\_limits=None*, *nodegroup=None*)

Bases: `Node`

Class which handles where the output goes. This can be any kind of file, e.g. image files, textfiles, config files, etc.

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_dataschemafile\_\_** = 'SinkNode.schema.json'

**\_\_getstate\_\_**()

Retrieve the state of the Node

**Returns**

the state of the object

**Rtype dict**

**\_\_init\_\_**(*datatype*, *id\_=None*, *parent=None*, *resource\_limits=None*, *nodegroup=None*)

Instantiation of the SourceNode.

**Parameters**

- **datatype** – The datatype of the output.
- **id** – the id of the node to create

**Returns**

newly created sink node

usage example:

```
>>> import fastr
>>> network = fastr.create_network()
>>> sink = network.create_sink(datatype=types['ITKImageFile'], id_='SinkN')
```

**\_\_module\_\_** = 'fastr.planning.node'

**\_\_setstate\_\_**(*state*)

Set the state of the Node by the given state.

**Parameters**

**state** (*dict*) – The state to populate the object with

**Returns**

None

**property datatype**

The datatype of the data this sink can store.

**draw**(*context, graph, color=None*)**property input**

The default input of the sink Node

```
class fastr.planning.node.SourceNode(datatype, id_=None, parent=None, resource_limits=None,
                                     nodegroup=None)
```

Bases: [FlowNode](#)

Class providing a connection to data resources. This can be any kind of file, stream, database, etc from which data can be received.

```
__abstractmethods__ = frozenset({})
```

```
__dataschemafile__ = 'SourceNode.schema.json'
```

```
__getstate__()
```

Retrieve the state of the SourceNode

**Returns**

the state of the object

**Rtype dict**

```
__init__(datatype, id_=None, parent=None, resource_limits=None, nodegroup=None)
```

Instantiation of the SourceNode.

**Parameters**

- **datatype** – The (id of) the datatype of the output.
- **id** – The url pattern.

This class should never be instantiated directly (unless you know what you are doing). Instead create a source using the network class like shown in the usage example below.

usage example:

```
>>> import fastr
>>> network = fastr.create_network()
>>> source = network.create_source(datatype=types['ITKImageFile'], id_='sourceN
↪')
```

```
__module__ = 'fastr.planning.node'
```

```
__setstate__(state)
```

Set the state of the SourceNode by the given state.

**Parameters****state** (*dict*) – The state to populate the object with**Returns**

None



**property datatype**

The datatype of the data this source supplies.

**property dimensions**

The dimensions in the SourceNode output. These will be reflected in the SampleIdLists.

**draw**(*context*, *graph*, *color=None*)

**property nodegroup****property output**

Shorthand for `self.outputs['output']`

**set\_data**(*data*, *ids=None*)

Set the data of this source node.

**Parameters**

- **data** (*dict*, *OrderedDict* or *list of urls*) – the data to use
- **ids** – if data is a list, a list of accompanying ids

**property sourcegroup****property valid**

This does nothing. It only overloads the valid method of Node(). The original is intended to check if the inputs are connected to some output. Since this class does not implement inputs, it is skipped.

## Subpackages

### test Package

#### test\_network Module

#### test\_node Module

### plugins Package

#### plugins Package

The plugins module holds all plugins loaded by Fastr. It is empty on start and gets filled by the BasePluginManager

**class** `fastr.plugins.BlockingExecution`(*finished\_callback=None*, *cancelled\_callback=None*)

Bases: `ExecutionPlugin`

The blocking execution plugin is a special plugin which is meant for debug purposes. It will not queue jobs but immediately execute them inline, effectively blocking fastr until the Job is finished. It is the simplest execution plugin and can be used as a template for new plugins or for testing purposes.

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_init\_\_**(*finished\_callback=None*, *cancelled\_callback=None*)

Setup the ExecutionPlugin

**Parameters**

- **finished\_callback** – the callback to call after a job finished

- **cancelled\_callback** – the callback to call after a job cancelled

**Returns**

newly created ExecutionPlugin

```
__module__ = 'fastr.plugins'
```

**cleanup()**

Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections, etc.

**Parameters**

**force** – force cleanup (e.g. kill instead of join a process)

```
filename =
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/executionplugins/blockingexecution.py'
```

```
module = <module 'blockingexecution' from
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/executionplugins/blockingexecution.py'>
```

**classmethod test()**

Test the plugin, default behaviour is just to instantiate the plugin

```
class fastr.plugins.CommaSeperatedValueFile
```

Bases: [IOPlugin](#)

The CommaSeperatedValueFile an expand-only type of IOPlugin. No URLs can actually be fetched, but it can expand a single URL into a larger amount of URLs.

The `csv://` URL is a `vfs://` URL with a number of query variables available. The URL mount and path should point to a valid CSV file. The query variable then specify what column(s) of the file should be used.

The following variable can be set in the query:

variable	usage
value	the column containing the value of interest, can be int for index or string for key
id	the column containing the sample id (optional)
header	indicates if the first row is considered the header, can be <code>true</code> or <code>false</code> (optional)
delimiter	the delimiter used in the csv file (optional)
quote	the quote character used in the csv file (optional)
reformat	a reformatting string so that <code>value = reformat.format(value)</code> (used before <code>relative_path</code> )
relative_path	indicates the entries are relative paths (for files), can be <code>true</code> or <code>false</code> (optional)

The header is by default `false` if the neither the `value` and `id` are set as a string. If either of these are a string, the header is required to define the column names and it automatically is assumed `true`

The delimiter and quota characters of the file should be detected automatically using the [Sniffer](#), but can be forced by setting them in the URL.

Example of valid csv URLs:

```
# Use the first column in the file (no header row assumed)
csv://mount/some/dir/file.csv?value=0
```

(continues on next page)

(continued from previous page)

```
# Use the images column in the file (first row is assumed header row)
csv://mount/some/dir/file.csv?value=images

# Use the segmentations column in the file (first row is assumed header row)
# and use the id column as the sample id
csv://mount/some/dir/file.csv?value=segmentations&id=id

# Use the first column as the id and the second column as the value
# and skip the first row (considered the header)
csv://mount/some/dir/file.csv?value=1&id=0&header=true

# Use the first column and force the delimiter to be a comma
csv://mount/some/dir/file.csv?value=0&delimiter=,
```

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
expand_url(url)
```

(abstract) Expand an URL. This allows a source to collect multiple samples from a single url. The URL will have a wildcard or point to something with info and multiple urls will be returned.

**Parameters**

**url** (*str*) – url to expand

**Returns**

the resulting url(s), a tuple if multiple, otherwise a str

**Return type**

*str* or *tuple* of *str*

```
filename =
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/ioplugins/commaseperatedvaluefile.py'
```

```
module = <module 'commaseperatedvaluefile' from
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/ioplugins/commaseperatedvaluefile.py'>
```

```
scheme = 'csv'
```

```
class fastr.plugins.CrossValidation
```

Bases: FlowPlugin

Advanced flow plugin that generated a cross-validation data flow. The node need an input with data and an input number of folds. Based on that the outputs test and train will be supplied with a number of data sets.

```
__abstractmethods__ = frozenset({})
```

```
__module__ = 'fastr.plugins'
```

```
static execute(payload)
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/  
python3.6/site-packages/fastr/resources/plugins/flowplugins/crossvalidation.py'
```

```
module = <module 'crossvalidation' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/  
site-packages/fastr/resources/plugins/flowplugins/crossvalidation.py'>
```

```
class fastr.plugins.DRMAAExecution(finished_callback=None, cancelled_callback=None)
```

Bases: [ExecutionPlugin](#)

A DRMAA execution plugin to execute Jobs on a Grid Engine cluster. It uses a configuration option for selecting the queue to submit to. It uses the python drmaa package.

---

**Note:** To use this plugin, make sure the drmaa package is installed and that the execution is started on an SGE submit host with DRMAA libraries installed.

---

---

**Note:** This plugin is at the moment tailored to SGE, but it should be fairly easy to make different subclasses for different DRMAA supporting systems.

---

**CANCELS\_DEPENDENCIES = False**

Indicates that when a job is cancelled the dependencies

```
GE_NATIVE_SPEC = {'DEPENDS': '-hold_jid {hold_list}', 'DEPENDS_SEP': ',',  
'ERRORLOG': '-e {errorlog}', 'HOLD': '-h', 'MEMORY': '-l h_vmem={memory}', 'NCORES':  
'-pe smp {ncores:d}', 'OUTPUTLOG': '-o {outputlog}', 'QUEUE': '-q {queue}',  
'WALLTIME': '-l h_rt={walltime}', 'WD': '-wd {workdir}'}
```

```
NATIVE_SPEC = {'grid_engine': {'DEPENDS': '-hold_jid {hold_list}', 'DEPENDS_SEP':  
,', 'ERRORLOG': '-e {errorlog}', 'HOLD': '-h', 'MEMORY': '-l h_vmem={memory}',  
'NCORES': '-pe smp {ncores:d}', 'OUTPUTLOG': '-o {outputlog}', 'QUEUE': '-q  
{queue}', 'WALLTIME': '-l h_rt={walltime}', 'WD': '-wd {workdir}'}, 'torque':  
{'CWD': '', 'DEPENDS': '-W depend=afterok:{hold_list}', 'DEPENDS_SEP': ':',  
'ERRORLOG': '-e {errorlog}', 'HOLD': '-h', 'MEMORY': '-l mem={memory}', 'NCORES':  
'-l procs={ncores:d}', 'OUTPUTLOG': '-o {outputlog}', 'QUEUE': '-q {queue}',  
'WALLTIME': '-l walltime={walltime}'}
```

**SUPPORTS\_CANCEL = True**

Indicates if the plugin can cancel queued jobs

**SUPPORTS\_DEPENDENCY = True**

Indicate if the plugin can manage job dependencies, if not the base plugin job dependency system will be used and jobs will only be submitted when all dependencies are met.

**SUPPORTS\_HOLD\_RELEASE = True**

Indicates if the plugin can queue jobs in a hold state and can release them again (if not, the base plugin will create a hidden queue for held jobs)

```
TORQUE_NATIVE_SPEC = {'CWD': '', 'DEPENDS': '-W depend=afterok:{hold_list}',  
'DEPENDS_SEP': ':', 'ERRORLOG': '-e {errorlog}', 'HOLD': '-h', 'MEMORY': '-l  
mem={memory}', 'NCORES': '-l procs={ncores:d}', 'OUTPUTLOG': '-o {outputlog}',  
'QUEUE': '-q {queue}', 'WALLTIME': '-l walltime={walltime}'}
```

```
__abstractmethods__ = frozenset({})
```

```
__init__(finished_callback=None, cancelled_callback=None)
```

Setup the ExecutionPlugin

#### Parameters

- **finished\_callback** – the callback to call after a job finished
- **cancelled\_callback** – the callback to call after a job cancelled

#### Returns

newly created ExecutionPlugin

```
__module__ = 'fastr.plugins'
```

```
check_threads()
```

Check if the threads are still alive, but make sure it is only done once per minute

```
cleanup()
```

Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections, etc.

#### Parameters

**force** – force cleanup (e.g. kill instead of join a process)

```
collect_jobs()
```

```
configuration_fields = {'drmaa_engine': (<class 'str'>, 'grid_engine', 'The engine
to use (options: grid_engine, torque)', 'drmaa_job_check_interval': (<class
'int'>, 900, 'The interval in which the job checker will start to check for stale
jobs'), 'drmaa_max_jobs': (<class 'int'>, 0, 'The maximum jobs that can be send to
the scheduler at the same time (0 for no limit)'), 'drmaa_num_undetermined_to_fail':
(<class 'int'>, 3, 'Number of consecutive times a job state has be undetermined to
be considered to have failed'), 'drmaa_queue': (<class 'str'>, 'week', 'The default
queue to use for jobs send to the scheduler')}
```

```
create_native_spec(queue, walltime, memory, ncores, outputLog, errorLog, hold_job, hold, work_dir)
```

Create the native spec for the DRMAA scheduler. Needs to be implemented in the subclasses

#### Parameters

- **queue** (*str*) – the queue to submit to
- **walltime** (*str*) – walltime specified
- **memory** (*str*) – memory requested
- **ncores** (*int*) – number of cores requested
- **outputLog** (*str*) – the location of the stdout log
- **errorLog** (*str*) – the location of stderr log
- **hold\_job** (*list*) – list of jobs to depend on
- **hold** (*bool*) – flag if job should be submitted in hold mode

#### Returns

```
dispatch_callbacks()
```

**ensure\_threads()**

Start thread if not defined, or restart if they somehow died accidentally

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/executionplugins/drmaaexecution.py'
```

```
module = <module 'drmaaexecution' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/executionplugins/drmaaexecution.py'>
```

**property n\_current\_jobs**

**regression\_check()**

**send\_job**(*command, arguments, work\_dir, queue=None, resources=None, job\_name=None,
joinLogFiles=False, outputLog=None, errorLog=None, hold\_job=None, hold=False*)

**property spec\_fields**

**submit\_jobs()**

**classmethod test()**

Test the plugin, default behaviour is just to instantiate the plugin

**class** fastr.plugins.**DockerTarget**(*binary, docker\_image*)

Bases: [Target](#)

A tool target that is located in a Docker images. Can be run using docker-py. A docker target only need two variables: the binary to call within the docker container, and the docker container to use.

```
{
  "arch": "*",
  "os": "*",
  "binary": "bin/test.py",
  "docker_image": "fastr/test"
}
```

```
<target os="*" arch="*" binary="bin/test.py" docker_image="fastr/test">
```

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_enter\_\_()**

Set the environment in such a way that the target will be on the path.

**\_\_exit\_\_**(*exc\_type, exc\_value, traceback*)

Cleanup the environment where needed

**\_\_init\_\_**(*binary, docker\_image*)

Define a new docker target.

**Parameters**

**docker\_image** (*str*) – Docker image to use

**\_\_module\_\_** = 'fastr.plugins'

**property container**

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/targetplugins/dockertarget.py'
```

```
module = <module 'dockertarget' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/targetplugins/dockertarget.py'>
```

**monitor\_docker**(*container*, *resources*)

Monitor a docker container and profile the cpu, memory and io use. Register the resource use every `_MONITOR_INTERVAL` seconds.

#### Parameters

- **container** (*ContainerCollection*) – process to monitor
- **resources** (*ProcessUsageCollection*) – list to append measurements to

**run\_command**(*command*)

Run a command with the target

#### Return type

*TargetResult*

**class** `fastr.plugins.ElasticsearchReporter`

Bases: *ReportingPlugin*

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_init\_\_**()

The BasePlugin constructor.

#### Returns

the created plugin

#### Return type

BasePlugin

#### Raises

*FastrPluginNotLoaded* – if the plugin did not load correctly

**\_\_module\_\_** = `'fastr.plugins'`

**activate**()

Activate the reporting plugin

```
configuration_fields = {'elasticsearch_debug': (<class 'bool'>, False, 'Setup
elasticsearch debug mode to send stdout stderr on job succes'),
'elasticsearch_host': (<class 'str'>, '', 'The elasticsearch host to report to'),
'elasticsearch_index': (<class 'str'>, 'fastr', 'The elasticsearch index to store
data in')}
```

**elasticsearch\_update\_status**(*job*)

**filename** =

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/reportingplugins/elasticsearchreporter.py'
```

**job\_updated**(*job*)

```
module = <module 'elasticsearchreporter' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/reportingplugins/elasticsearchreporter.py'>
```

```
classmethod test()
```

Test the plugin, default behaviour is just to instantiate the plugin

```
class fastr.plugins.FastrInterface(id_, document)
```

Bases: [Interface](#)

The default Interface for fastr. For the command-line Tools as used by fastr. It build a commandline call based on the input/output specification.

The fields that can be set in the interface:

Attribute		Description
id		The id of this Tool (used internally in fastr)
inputs[]		List of Inputs that can are accepted by the Tool
	id	ID of the Input
	name	Longer name of the Input (more human readable)
	datatype	The ID of the DataType of the Input <sup>1</sup>
	enum[]	List of possible values for an EnumType (created on the fly by fastr) <sup>Page 261, 1</sup>
	prefix	Commandline prefix of the Input (e.g. -in, -i)
	cardinality	Cardinality of the Input
	repeat_prefix	Flag indicating if for every value of the Input the prefix is repeated
	required	Flag indicating if the input is required
	nospace	Flag indicating if there is no space between prefix and value (e.g. -in=val)
	format	For DataTypes that have multiple representations, indicate which one to use
	default	Default value for the Input
	description	Long description for an input
outputs[]		List of Outputs that are generated by the Tool (and accessible to fastr)
	id	ID of the Output
	name	Longer name of the Output (more human readable)
	datatype	The ID of the DataType of the Output <sup>Page 261, 1</sup>
	enum[]	List of possible values for an EnumType (created on the fly by fastr) <sup>Page 261, 1</sup>

continues on next page



Table 3.1 – continued from previous page

Attribute		Description
	<code>prefix</code>	Commandline prefix of the Output (e.g. <code>-out, -o</code> )
	<code>cardinality</code>	Cardinality of the Output
	<code>repeat_prefix</code>	Flag indicating if for every value of the Output the prefix is repeated
	<code>required</code>	Flag indicating if the input is required
	<code>nospace</code>	Flag indicating if there is no space between prefix and value (e.g. <code>-out=val</code> )
	<code>format</code>	For DataTypes that have multiple representations, indicate which one to use
	<code>description</code>	Long description for an input
	<code>action</code>	Special action (defined per DataType) that needs to be performed before creating output value (e.g. <code>'ensure'</code> will make sure an output directory exists)
	<code>automatic</code>	Indicate that output doesn't require commandline argument, but is created automatically by a Tool <sup>2</sup>
	<code>method</code>	The collector plugin to use for the gathering automatic output, see the <a href="#">Collector plugins</a>
	<code>location</code>	Definition where to an automatically, usage depends on the <code>method</code> <sup>Page 261, 2</sup>

```
__abstractmethods__ = frozenset({})
```

```
__dataschemafile__ = 'FastrInterface.schema.json'
```

```
__eq__(other)
```

Return `self==value`.

```
__getstate__()
```

Get the state of the `FastrInterface` object.

**Returns**

state of interface

**Return type**

`dict`

```
__hash__ = None
```

```
__init__(id_, document)
```

The `BasePlugin` constructor.

**Returns**

the created plugin

<sup>1</sup> datatype and enum are conflicting entries, if both specified datatype has presedence

<sup>2</sup> More details on defining automatica output are given in [TODO]

**Return type**

BasePlugin

**Raises**

*FastrPluginNotLoaded* – if the plugin did not load correctly

**\_\_module\_\_** = 'fastr.plugins'

**\_\_setstate\_\_**(*state*)

Set the state of the Interface

**check\_input\_id**(*id\_*)

Check if an id for an object is valid and unused in the Tool. The method will always returns True if it does not raise an exception.

**Parameters**

**id** (*str*) – the id to check

**Returns**

True

**Raises**

- *FastrValueError* – if the id is not correctly formatted
- *FastrValueError* – if the id is already in use

**check\_output\_id**(*id\_*)

Check if an id for an object is valid and unused in the Tool. The method will always returns True if it does not raise an exception.

**Parameters**

**id** (*str*) – the id to check

**Returns**

True

**Raises**

- *FastrValueError* – if the id is not correctly formatted
- *FastrValueError* – if the id is already in use

**static collect\_errors**(*result*)

Special error collection for fastr interfaces

**collect\_results**(*result*)

Collect all results of the interface

**collector\_plugin\_type**

alias of CollectorPlugin

```
collectors = CollectorPluginManager [37m[42m[1mLoaded[0m json : <CollectorPlugin:
JsonCollector> [37m[42m[1mLoaded[0m path : <CollectorPlugin: PathCollector>
[37m[42m[1mLoaded[0m stdout : <CollectorPlugin: StdoutCollector>
```

**execute**(*target, payload*)

Execute the interface using a specific target and payload (containing a set of values for the arguments)

**Parameters**

- **target** (*SampleId*) – the target to use

- **payload** (*dict*) – the values for the arguments

**Returns**

result of the execution

**Return type**

*InterfaceResult*

**property expanding**

Indicates whether or not this Interface will result in multiple samples per run. If the flow is unaffected, this will be zero, if it is nonzero it means that number of dimension will be added to the sample array.

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/interfaceplugins/fastrinterface.py'
```

**get\_arguments**(*values*)

Get the argument list for this interface

**Returns**

return list of arguments

**get\_command**(*target*, *payload*)**get\_specials**(*payload*, *output*, *cardinality\_nr*)

Get special attributes. Returns tuples for specials, inputs and outputs that are used for formatting substitutions.

**Parameters**

- **output** – Output for which to get the specials
- **cardinality\_nr** (*int*) – the cardinality number

**property inputs**

OrderedDict of Inputs connected to the Interface. The format should be {input\_id: InputSpec}.

```
module = <module 'fastrinterface' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/interfaceplugins/fastrinterface.py'>
```

**property outputs**

OrderedDict of Output connected to the Interface. The format should be {output\_id: OutputSpec}.

**class** fastr.plugins.FileSystem

Bases: *IOPlugin*

The FileSystem plugin is create to handle `file://` type or URLs. This is generally not a good practice, as this is not portable over between machines. However, for test purposes it might be useful.

The URL scheme is rather simple: `file://host/path` (see [wikipedia](#) for details)

We do not make use of the `host` part and at the moment only support localhost (just leave the host empty) leading to `file:///` URLs.

**Warning:** This plugin ignores the hostname in the URL and does only accept driver letters on Windows in the form `c:/`

```
__abstractmethods__ = frozenset({})
```

**`__init__()`**

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

**`__module__`** = `'fastr.plugins'`

**`fetch_url(inurl, outpath)`**

Fetch the files from the file.

**Parameters**

- **`inurl`** – url to the item in the data store, starts with `file://`
- **`outpath`** – path where to store the fetch data locally

**`fetch_value(inurl)`**

Fetch a value from an external file file.

**Parameters**

**`inurl`** – url of the value to read

**Returns**

the fetched value

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/ioplugins/filesystem.py'
```

```
module = <module 'filesystem' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/  
site-packages/fastr/resources/plugins/ioplugins/filesystem.py'>
```

**`path_to_url(path, mountpoint=None)`**

Construct an url from a given mount point and a relative path to the mount point.

**`put_url(inpath, outurl)`**

Put the files to the external data store.

**Parameters**

- **`inpath`** – path of the local data
- **`outurl`** – url to where to store the data, starts with `file://`

**`put_value(value, outurl)`**

Put the value in the external data store.

**Parameters**

- **`value`** – value to store
- **`outurl`** – url to where to store the data, starts with `file://`

**`scheme`** = `'file'`

**`url_to_path(url)`**

Get the path to a file from a url. Currently supports the `file://` scheme

Examples:

```
>>> 'file:///d:/data/project/file.ext'  
'd:\data\project\file.ext'
```

**Warning:** `file://` will not function cross platform and is mainly for testing

**class** `fastr.plugins.FlowInterface(id_, document)`

Bases: `Interface`

The Interface use for AdvancedFlowNodes to create the advanced data flows that are not implemented in the fastr. This allows nodes to implement new data flows using the plugin system.

The definition of FlowInterfaces are very similar to the default FastrInterfaces.

---

**Note:** A flow interface should be using a specific FlowPlugin

---

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_dataschemafile\_\_** = `'FastrInterface.schema.json'`

**\_\_eq\_\_**(*other*)

Return self==value.

**\_\_getstate\_\_**()

Get the state of the FastrInterface object.

**Returns**

state of interface

**Return type**

`dict`

**\_\_hash\_\_** = `None`

**\_\_init\_\_**(*id\_, document*)

The BasePlugin constructor.

**Returns**

the created plugin

**Return type**

`BasePlugin`

**Raises**

`FastrPluginNotLoaded` – if the plugin did not load correctly

**\_\_module\_\_** = `'fastr.plugins'`

**\_\_setstate\_\_**(*state*)

Set the state of the Interface

**execute**(*target, payload*)

Execute the interface given the a target and payload. The payload should have the form:

```
{
  'input': {
    'input_id_a': (value, value),
    'input_id_b': (value, value)
  },
  'output': {
```

(continues on next page)

(continued from previous page)

```

    'output_id_a': (value, value),
    'output_id_b': (value, value)
}
}

```

**Parameters**

- **target** – the target to call
- **payload** – the payload to use

**Returns**

the result of the execution

**Return type**

(tuple of) *InterfaceResult*

**property expanding**

Indicates whether or not this Interface will result in multiple samples per run. If the flow is unaffected, this will be zero, if it is nonzero it means that number of dimension will be added to the sample array.

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/interfaceplugins/flowinterface.py'
```

**flow\_plugin\_type**

alias of FlowPlugin

```
flow_plugins = FlowPluginManager [37m[42m[1mLoaded[0m CrossValidation :
<FlowPlugin: CrossValidation>
```

**property inputs**

OrderedDict of Inputs connected to the Interface. The format should be {input\_id: InputSpec}.

```
module = <module 'flowinterface' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/interfaceplugins/flowinterface.py'>
```

**property outputs**

OrderedDict of Output connected to the Interface. The format should be {output\_id: OutputSpec}.

```
class fastr.plugins.HTTPPlugin
```

Bases: *IOPlugin*

**Warning:** This Plugin is still under development and has not been tested at all. example url: <https://server.io/path/to/resource>

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
fetch_url(inurl, outpath)
```

Download file from server.

#### Parameters

- **inurl** – url to the file.
- **outpath** – path to store file

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/ioplugins/httpplugin.py'
```

```
module = <module 'httpplugin' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/ioplugins/httpplugin.py'>
```

```
scheme = ('https', 'http')
```

```
class fastr.plugins.LinearExecution(finished_callback=None, cancelled_callback=None)
```

Bases: [ExecutionPlugin](#)

An execution engine that has a background thread that executes the jobs in order. The queue is a simple FIFO queue and there is one worker thread that operates in the background. This plugin is meant as a fallback when other plugins do not function properly. It does not multi-processing so it is safe to use in environments that do not support that.

```
__abstractmethods__ = frozenset({})
```

```
__init__(finished_callback=None, cancelled_callback=None)
```

Setup the ExecutionPlugin

#### Parameters

- **finished\_callback** – the callback to call after a job finished
- **cancelled\_callback** – the callback to call after a job cancelled

#### Returns

newly created ExecutionPlugin

```
__module__ = 'fastr.plugins'
```

```
cleanup()
```

Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections, etc.

#### Parameters

- **force** – force cleanup (e.g. kill instead of join a process)

```
exec_worker()
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/executionplugins/linearexecution.py'
```

```
module = <module 'linearexecution' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/executionplugins/linearexecution.py'>
```

**classmethod** `test()`

Test the plugin, default behaviour is just to instantiate the plugin

**class** `fastr.plugins.LocalBinaryTarget(binary, paths=None, environment_variables=None, initscripts=None, modules=None, interpreter=None, **kwargs)`

Bases: *SubprocessBasedTarget*

A tool target that is a local binary on the system. Can be found using environmentmodules or a path on the executing machine. A local binary target has a number of fields that can be supplied:

- **binary** (required): the name of the binary/script to call, can also be called `bin` for backwards compatibility.
- **modules**: list of modules to load, this can be environmentmodules or lmod modules. If modules are given, the `paths`, `environment_variables` and `initscripts` are ignored.
- **paths**: a list of paths to add following the structure `{"value": "/path/to/dir", "type": "bin"}`. The types can be `bin` if the it should be added to `$PATH` or `lib` if it should be added to te library path (e.g. `$LD_LIBRARY_PATH` for linux).
- **environment\_variables**: a dictionary of environment variables to set.
- **initscript**: a list of script to run before running the main tool
- **interpreter**: the interpreter to use to call the binary e.g. `python`

The LocalBinaryTarget will first check if there are modules given and the module subsystem is loaded. If that is the case it will simply unload all current modules and load the given modules. If not it will try to set up the environment itself by using the following steps:

1. Prepend the bin paths to `$PATH`
2. Prepend the lib paths to the correct environment variable
3. Setting the other environment variables given (`$PATH` and the system library path are ignored and cannot be set that way)
4. Call the initscripts one by one

The definition of the target in JSON is very straightforward:

```
{
  "binary": "bin/test.py",
  "interpreter": "python",
  "paths": [
    {
      "type": "bin",
      "value": "vfs://apps/test/bin"
    },
    {
      "type": "lib",
      "value": "./lib"
    }
  ],
  "environment_variables": {
    "othervar": 42,
    "short_var": 1,
    "testvar": "value1"
  },
  "initscripts": [
```

(continues on next page)



(continued from previous page)

```

    "bin/init.sh"
],
"modules": ["elastix/4.8"]
}

```

In XML the definition would be in the form of:

```

<target os="linux" arch="*" modules="elastix/4.8" bin="bin/test.py" interpreter=
→ "python">
  <paths>
    <path type="bin" value="vfs://apps/test/bin" />
    <path type="lib" value="./lib" />
  </paths>
  <environment_variables short_var="1">
    <testvar>value1</testvar>
    <othervar>42</othervar>
  </environment_variables>
  <initscripts>
    <initscript>bin/init.sh</initscript>
  </initscripts>
</target>

```

```

DYNAMIC_LIBRARY_PATH_DICT = {'darwin': 'DYLD_LIBRARY_PATH', 'linux':
'LD_LIBRARY_PATH', 'windows': 'PATH'}

```

```
__abstractmethods__ = frozenset({})
```

```
__enter__()
```

Set the environment in such a way that the target will be on the path.

```
__exit__(exc_type, exc_value, traceback)
```

Cleanup the environment

```
__init__(binary, paths=None, environment_variables=None, initscripts=None, modules=None,
         interpreter=None, **kwargs)
```

Define a new local binary target. Must be defined either using paths and optionally environment\_variables and initscripts, or environment modules.

```
__module__ = 'fastr.plugins'
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/targetplugins/localbinarytarget.py'
```

```
module = <module 'localbinarytarget' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/targetplugins/localbinarytarget.py'>
```

property paths

```
run_command(command)
```

Run a command with the target

Return type

*TargetResult*

```
class fastr.plugins.MacroTarget(network_file, method=None, function='main')
```

Bases: [Target](#)

A target for MacroNodes. This target cannot be executed as the MacroNode handles execution differently. But this contains the information for the MacroNode to find the internal Network.

```
__abstractmethods__ = frozenset({})
```

```
__init__(network_file, method=None, function='main')
```

Define a new local binary target. Must be defined either using paths and optionally environment\_variables and initscripts, or enviroment modules.

```
__module__ = 'fastr.plugins'
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/targetplugins/macrotarget.py'
```

```
module = <module 'macrotarget' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/targetplugins/macrotarget.py'>
```

```
run_command(command)
```

Run a command with the target

```
classmethod test()
```

Test if singularity is availble on the path

```
class fastr.plugins.NetworkScope
```

Bases: [IOPlugin](#)

A simple source plugin that allows to get data from the Network scope. This uses the network:// scheme.

An uri of network://atlases/image\_01.nii.gz would be translated to vfs://mount/network/atlas/image\_01.nii.gz given that the network would be created/loaded from vfs://mount/network/networkfile.py.

**Warning:** This means that the network file must be present in a folder mounted in the vfs system. Fastr will use a vfs to translate the path between main process and execution workers.

If the resulting uri should be a different vfs-based url that the default vfs:// then a combined scheme can be used. For example network+vfslist://atlases/list.txt would be translated into vfslist://mount/network/atlas/list.txt and the result would be run by the vfslist plugin.

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/ioplugins/networkscope.py'
```

```
module = <module 'networkscope' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/ioplugins/networkscope.py'>
```

```
scheme = ('network', 'network+')
```

```
class fastr.plugins.NipypeInterface(id_, nipype_cls=None, document=None)
```

Bases: [Interface](#)

Experimental interfaces to using nipype interfaces directly in fastr tools, only using a simple reference.

To create a tool using a nipype interface just create an interface with the correct type and set the nipype argument to the correct class. For example in an xml tool this would become:

```
<interface class="NipypeInterface">
  <nipype_class>nipype.interfaces.elastix.Registration</nipype_class>
</interface>
```

---

**Note:** To use these interfaces nipype should be installed on the system.

---

**Warning:** This interface plugin is basically functional, but highly experimental!

```
__abstractmethods__ = frozenset({})
```

```
__eq__(other)
```

Return self==value.

```
__getstate__()
```

Retrieve the state of the Interface

**Returns**

the state of the object

**Rtype dict**

```
__hash__ = None
```

```
__init__(id_, nipype_cls=None, document=None)
```

The BasePlugin constructor.

**Returns**

the created plugin

**Return type**

BasePlugin

**Raises**

[FastrPluginNotLoaded](#) – if the plugin did not load correctly

```
__module__ = 'fastr.plugins'
```

```
__setstate__(state)
```

Set the state of the Interface

**execute**(*target*, *payload*)

Execute the interface using a specific target and payload (containing a set of values for the arguments)

**Parameters**

- **target** (*SampleId*) – the target to use
- **payload** (*dict*) – the values for the arguments

**Returns**

result of the execution

**Return type**

*InterfaceResult*

**property expanding**

Indicates whether or not this Interface will result in multiple samples per run. If the flow is unaffected, this will be zero, if it is nonzero it means that number of dimension will be added to the sample array.

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/interfaceplugins/nipypeinterface.py'
```

**get\_type**(*trait*)

**property inputs**

OrderedDict of Inputs connected to the Interface. The format should be {input\_id: InputSpec}.

```
module = <module 'nipypeinterface' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/  
site-packages/fastr/resources/plugins/interfaceplugins/nipypeinterface.py'>
```

**property outputs**

OrderedDict of Output connected to the Interface. The format should be {output\_id: OutputSpec}.

**classmethod test()**

Test the plugin, interfaces do not need to be tested on import

**class** fastr.plugins.Null

Bases: *IOPlugin*

The Null plugin is create to handle null:// type or URLs. These URLs are indicating the sink should not do anything. The data is not written to anywhere. Besides the scheme, the rest of the URL is ignored.

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/  
python3.6/site-packages/fastr/resources/plugins/ioplugins/null.py'
```

```
module = <module 'null' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/  
site-packages/fastr/resources/plugins/ioplugins/null.py'>
```

**put\_url**(*inpath*, *outurl*)

Put the files to the external data store.

**Parameters**

- **inpath** – path of the local data
- **outurl** – url to where to store the data, starts with file://

**put\_value**(*value*, *outurl*)

Put the value in the external data store.

**Parameters**

- **value** – value to store
- **outurl** – url to where to store the data, starts with file://

**scheme** = 'null'

**class** fastr.plugins.PimReporter

Bases: [ReportingPlugin](#)

**SUPPORTED\_APIS** = {2: <class 'pimreporter.PimAPIv2'>}

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_init\_\_**()

The BasePlugin constructor.

**Returns**

the created plugin

**Return type**

BasePlugin

**Raises**

[FastrPluginNotLoaded](#) – if the plugin did not load correctly

**\_\_module\_\_** = 'fastr.plugins'

**activate**()

Activate the reporting plugin

**configuration\_fields** = {'pim\_batch\_size': (<class 'int'>, 100, 'Maximum number of jobs that can be send to PIM in a single interval'), 'pim\_debug': (<class 'bool'>, False, 'Setup PIM debug mode to send stdout stderr on job success'), 'pim\_finished\_timeout': (<class 'int'>, 10, 'Maximum number of seconds after the network finished in which PIM tries to synchronize all remaining jobs'), 'pim\_host': (<class 'str'>, '', 'The PIM host to report to'), 'pim\_update\_interval': (<class 'float'>, 2.5, 'The interval in which to send jobs to PIM'), 'pim\_username': (<class 'str'>, 'docs', 'Username to send to PIM', 'Username of the currently logged in user')}

**filename** = '/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/reportingplugins/pimreporter.py'

**job\_updated**(*job*)

**log\_record\_emitted**(*record*)

```
module = <module 'pimreporter' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/reportingplugins/pimreporter.py'>
```

```
run_finished(run)
```

```
run_started(run)
```

```
class fastr.plugins.ProcessPoolExecution(finished_callback=None, cancelled_callback=None,
                                         nr_of_workers=None)
```

Bases: [ExecutionPlugin](#)

A local execution plugin that uses multiprocessing to create a pool of worker processes. This allows fastr to execute jobs in parallel with true concurrency. The number of workers can be specified in the fastr configuration, but the default amount is the number of cores - 1 with a minimum of 1.

**Warning:** The ProcessPoolExecution does not check memory requirements of jobs and running many workers might lead to memory starvation and thus an unresponsive system.

```
__abstractmethods__ = frozenset({})
```

```
__init__(finished_callback=None, cancelled_callback=None, nr_of_workers=None)
```

Setup the ExecutionPlugin

#### Parameters

- **finished\_callback** – the callback to call after a job finished
- **cancelled\_callback** – the callback to call after a job cancelled

#### Returns

newly created ExecutionPlugin

```
__module__ = 'fastr.plugins'
```

```
cleanup()
```

Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections, etc.

#### Parameters

**force** – force cleanup (e.g. kill instead of join a process)

```
configuration_fields = {'process_pool_worker_number': (<class 'int'>, 1, 'Number of
workers to use in a process pool')}
```

```
filename =
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/executionplugins/processpoolexecution.py'
```

```
job_finished_callback(result)
```

Receiver for the callback, it will split the result tuple and call job\_finished

#### Parameters

**result** (*tuple*) – return value of run\_job

```
module = <module 'processpoolexecution' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/executionplugins/processpoolexecution.py'>
```

```
classmethod test()
```

Test the plugin, default behaviour is just to instantiate the plugin

```
class fastr.plugins.RQExecution(finished_callback=None, cancelled_callback=None)
```

Bases: [ExecutionPlugin](#)

A execution plugin based on Redis Queue. Fastr will submit jobs to the redis queue and workers will peel the jobs from the queue and process them.

This system requires a running redis database and the database url has to be set in the fastr configuration.

---

**Note:** This execution plugin required the `redis` and `rq` packages to be installed before it can be loaded properly.

---

```
__abstractmethods__ = frozenset({})
```

```
__init__(finished_callback=None, cancelled_callback=None)
```

Setup the ExecutionPlugin

#### Parameters

- **finished\_callback** – the callback to call after a job finished
- **cancelled\_callback** – the callback to call after a job cancelled

#### Returns

newly created ExecutionPlugin

```
__module__ = 'fastr.plugins'
```

```
check_finished()
```

```
cleanup()
```

Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections, etc.

#### Parameters

**force** – force cleanup (e.g. kill instead of join a process)

```
configuration_fields = {'rq_host': (<class 'str'>, 'redis://localhost:6379/0', 'The url of the redis serving the redis queue'), 'rq_queue': (<class 'str'>, 'default', 'The redis queue to use')}
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/executionplugins/rqexecution.py'
```

```
module = <module 'rqexecution' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/executionplugins/rqexecution.py'>
```

```
classmethod run_job(job_id, job_command, job_stdout, job_stderr)
```

```
classmethod test()
```

Test the plugin, default behaviour is just to instantiate the plugin

```
class fastr.plugins.Reference
```

Bases: [IOPlugin](#)

The Reference plugin is create to handle `ref://` type or URLs. These URLs are to make the sink just write a simple reference file to the data. The reference file contains the `DataType` and the value so the result can be

reconstructed. It for files just leaves the data on disk by reference. This plugin is not useful for production, but is used for testing purposes.

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/ioplugins/reference.py'
```

```
module = <module 'reference' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/ioplugins/reference.py'>
```

```
push_sink_data(value, outurl, datatype=None)
```

Write out the sink data from the inpath to the outurl.

**Parameters**

- **value** (*str*) – the path of the data to be pushed
- **outurl** (*str*) – the url to write the data to
- **datatype** (*DataType*) – the datatype of the data, used for determining the total contents of the transfer

**Returns**

None

```
scheme = 'ref'
```

```
class fastr.plugins.S3Filesystem
```

Bases: *IOPlugin*

**Warning:** As this IOPlugin is under development, it has not been thoroughly tested.

example url: s3://bucket.server/path/to/resource

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
cleanup()
```

(abstract) Clean up the IOPlugin. This is to do things like closing files or connections. Will be called when the plugin is no longer required.



**expand\_url(url)**

Expand an S3 URL. This allows a source to collect multiple samples from a single url.

**Parameters**

**url** (*str*) – url to expand

**Returns**

the resulting url(s), a tuple if multiple, otherwise a str

**Return type**

*str* or *tuple* of *str*

**fetch\_url(inurl, outpath)**

Get the file(s) or values from s3.

**Parameters**

- **inurl** – url to the item in the data store
- **outpath** – path where to store the fetch data locally

**fetch\_value(inurl)**

Fetch a value from S3

**Parameters**

**inurl** – url of the value to read

**Returns**

the fetched value

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/ioplugins/s3filesystem.py'
```

```
module = <module 's3filesystem' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/ioplugins/s3filesystem.py'>
```

**put\_url(inpath, outurl)**

Upload the files to the S3 storage

**Parameters**

- **inpath** – path to the local data
- **outurl** – url to where to store the data in the external data store.

**put\_value(value, outurl)**

Put the value in S3

**Parameters**

- **value** – value to store
- **outurl** – url to where to store the data, starts with file://

```
scheme = ('s3', 's3list')
```

**classmethod test()**

Test the plugin, default behaviour is just to instantiate the plugin

```
class fastr.plugins.SimpleReport
```

Bases: [ReportingPlugin](#)

```
__abstractmethods__ = frozenset({})

__module__ = 'fastr.plugins'

filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/reportingplugins/simplereport.py'

module = <module 'simplereport' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/reportingplugins/simplereport.py'>

run_finished(run)
```

```
class fastr.plugins.SingularityTarget(binary, container, interpreter=None)
```

Bases: [SubprocessBasedTarget](#)

A tool target that is run using a singularity container, see the [singularity website](#)

- **binary** (required): the name of the binary/script to call, can also be called bin for backwards compatibility.
- **container** (required): the singularity container to run, this can be in url form for singularity pull or as a path to a local container
- **interpreter**: the interpreter to use to call the binary e.g. python

```
SINGULARITY_BIN = 'singularity'
```

```
__abstractmethods__ = frozenset({})
```

```
__enter__()
```

Set the environment in such a way that the target will be on the path.

```
__exit__(exc_type, exc_value, traceback)
```

Cleanup the environment

```
__init__(binary, container, interpreter=None)
```

Define a new local binary target. Must be defined either using paths and optionally environment\_variables and initscripts, or enviroment modules.

```
__module__ = 'fastr.plugins'
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/targetplugins/singularitytarget.py'
```

```
module = <module 'singularitytarget' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/targetplugins/singularitytarget.py'>
```

```
run_command(command)
```

Run a command with the target

```
classmethod test()
```

Test if singularity is availble on the path

```
class fastr.plugins.SlurmExecution(finished_callback=None, cancelled_callback=None)
```

Bases: [ExecutionPlugin](#)

The SlurmExecution plugin allows you to send the jobs to SLURM using the sbatch command. It is pure python and uses the sbatch, scancel, squeue and scontrol programs to control the SLURM scheduler.

```

SBATCH = 'sbatch'

SCANCEL = 'scancel'

SCONTROL = 'scontrol'

SQUEUE = 'squeue'

SQUEUE_FORMAT = '{"id": %.18i, "status": "%.2t"}'

STATUS_MAPPING = {'F': JobState.failed, 'R': JobState.running, 'CA':
JobState.cancelled, 'CD': JobState.finished, 'CF': JobState.running, 'CG':
JobState.running, 'NF': JobState.failed, 'PD': JobState.queued, 'RV':
JobState.cancelled, 'SE': JobState.failed, 'TO': JobState.queued}

SUPPORTS_CANCEL = True
    Indicates if the plugin can cancel queued jobs

SUPPORTS_DEPENDENCY = True
    Indicate if the plugin can manage job dependencies, if not the base plugin job dependency system will be
    used and jobs with only be submitted when all dependencies are met.

SUPPORTS_HOLD_RELEASE = True
    Indicates if the plugin can queue jobs in a hold state and can release them again (if not, the base plugin will
    create a hidden queue for held jobs)

__abstractmethods__ = frozenset({})

__init__(finished_callback=None, cancelled_callback=None)
    Setup the ExecutionPlugin

    Parameters
        • finished_callback – the callback to call after a job finished
        • cancelled_callback – the callback to call after a job cancelled

    Returns
        newly created ExecutionPlugin

__module__ = 'fastr.plugins'

cleanup()
    Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections,
    etc.

    Parameters
        force – force cleanup (e.g. kill instead of join a process)

configuration_fields = {'slurm_job_check_interval': (<class 'int'>, 30, 'The
interval in which the job checker will startto check for stale jobs'),
'slurm_partition': (<class 'str'>, '', 'The slurm partition to use')}

filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/executionplugins/slurmexecution.py'

job_status_check()

```

```
module = <module 'slurmexecution' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/executionplugins/slurmexecution.py'>
```

```
classmethod test()
```

Test the plugin, default behaviour is just to instantiate the plugin

```
class fastr.plugins.StrongrExecution(finished_callback=None, cancelled_callback=None)
```

Bases: [ExecutionPlugin](#)

```
__abstractmethods__ = frozenset({})
```

```
__init__(finished_callback=None, cancelled_callback=None)
```

Setup the ExecutionPlugin

#### Parameters

- **finished\_callback** – the callback to call after a job finished
- **cancelled\_callback** – the callback to call after a job cancelled

#### Returns

newly created ExecutionPlugin

```
__module__ = 'fastr.plugins'
```

```
check_finished()
```

```
cleanup()
```

Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections, etc.

#### Parameters

**force** – force cleanup (e.g. kill instead of join a process)

```
configuration_fields = {}
```

```
filename =
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/executionplugins/strongrexecution.py'
```

```
module = <module 'strongrexecution' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/executionplugins/strongrexecution.py'>
```

```
classmethod test()
```

Test the plugin, default behaviour is just to instantiate the plugin

```
class fastr.plugins.VirtualFileSystem
```

Bases: [VirtualFileSystem](#), [IOPlugin](#)

The virtual file system class. This is an IOPlugin, but also heavily used internally in fastr for working with directories. The VirtualFileSystem uses the `vfs://` url scheme.

A typical virtual filesystem url is formatted as `vfs://mountpoint/relative/dir/from/mount.ext`

Where the mountpoint is defined in the [Config file](#). A list of the currently known mountpoints can be found in the `fastr.config` object

```
>>> fastr.config.mounts
{'example_data': '/home/username/fastr-feature-documentation/fastr/fastr/examples/
↳data',
 'home': '/home/username/',
 'tmp': '/home/username/FastrTemp'}
```

This shows that a url with the mount home such as `vfs://home/tempdir/testfile.txt` would be translated into `/home/username/tempdir/testfile.txt`.

There are a few default mount points defined by Fastr (that can be changed via the config file).

mountpoint	default location
home	the users home directory ( <code>expanduser('~')</code> )
tmp	the fastr temporary dir, defaults to <code>tempfile.gettempdir()</code>
example_data	the fastr example data directory, defaults <code>\$FASTRDIR/example/data</code>

```
__abstractmethods__ = frozenset({})
```

```
__module__ = 'fastr.plugins'
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/ioplugins/virtualfilesystem.py'
```

```
module = <module 'virtualfilesystem' from
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/ioplugins/virtualfilesystem.py'>
```

```
scheme = 'vfs'
```

```
class fastr.plugins.VirtualFileSystemRegularExpression
```

Bases: [IOPlugin](#)

The `VirtualFileSystemValueList` an expand-only type of `IOPlugin`. No URLs can actually be fetched, but it can expand a single URL into a larger amount of URLs.

A `vfsregex://` URL is a `vfs` URL that can contain regular expressions on every level of the path. The regular expressions follow the [re module](#) definitions.

An example of a valid URLs would be:

```
vfsregex://tmp/network_dir/.*/*/__fastr_result__.pickle.gz
vfsregex://tmp/network_dir/nodeX/(?P<id>.*).__fastr_result__.pickle.gz
```

The first URL would result in all the `__fastr_result__.pickle.gz` in the working directory of a Network. The second URL would only result in the file for a specific node (nodeX), but by adding the named group id using `(?P<id>.*)` the sample id of the data is automatically set to that group (see [Regular Expression Syntax](#) under the special characters for more info on named groups in regular expression).

Concretely if we would have a directory `vfs://mount/somedir` containing:

```
image_1/Image.nii
image_2/image.nii
image_3/anotherimage.nii
image_5/inconsistentnamingftw.nii
```

we could match these files using `vfsregex://mount/somedir/(?P<id>image_\d+)/.*\..nii` which would result in the following source data after expanding the URL:

```
{'image_1': 'vfs://mount/somedir/image_1/Image.nii',
 'image_2': 'vfs://mount/somedir/image_2/image.nii',
 'image_3': 'vfs://mount/somedir/image_3/anotherimage.nii',
 'image_5': 'vfs://mount/somedir/image_5/inconsistentnamingftw.nii'}
```

Showing the power of this regular expression filtering. Also it shows how the ID group from the URL can be used to have sensible sample ids.

**Warning:** due to the nature of regexp on multiple levels, this method can be slow when having many matches on the lower level of the path (because the tree of potential matches grows) or when directories that are parts of the path are very large.

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
expand_url(url)
```

(abstract) Expand an URL. This allows a source to collect multiple samples from a single url. The URL will have a wildcard or point to something with info and multiple urls will be returned.

**Parameters**

**url** (*str*) – url to expand

**Returns**

the resulting url(s), a tuple if multiple, otherwise a str

**Return type**

*str* or *tuple* of *str*

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/
python3.6/site-packages/fastr/resources/plugins/ioplugins/
virtualfilesystemregularexpression.py'
```

```
module = <module 'virtualfilesystemregularexpression' from '/home/docs/checkouts/
readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/
resources/plugins/ioplugins/virtualfilesystemregularexpression.py'>
```

```
scheme = 'vfsregex'
```

```
class fastr.plugins.VirtualFileSystemValueList
```

Bases: *IOPlugin*

The VirtualFileSystemValueList an expand-only type of IOPlugin. No URLs can actually be fetched, but it can expand a single URL into a larger amount of URLs. A `vfslist://` URL basically is a url that points to a file using vfs. This file then contains a number lines each containing another URL.

If the contents of a file `vfs://mount/some/path/contents` would be:

```
vfs://mount/some/path/file1.txt
vfs://mount/some/path/file2.txt
vfs://mount/some/path/file3.txt
vfs://mount/some/path/file4.txt
```

Then using the URL `vfslist://mount/some/path/contents` as source data would result in the four files being pulled.

**Note:** The URLs in a `vfslist` file do not have to use the `vfs` scheme, but can use any scheme known to the `Fastr` system.

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the `IOPlugin`

**Returns**

newly created `IOPlugin`

```
__module__ = 'fastr.plugins'
```

```
expand_url(url)
```

(abstract) Expand an URL. This allows a source to collect multiple samples from a single url. The URL will have a wildcard or point to something with info and multiple urls will be returned.

**Parameters**

`url` (*str*) – url to expand

**Returns**

the resulting url(s), a tuple if multiple, otherwise a str

**Return type**

*str* or *tuple* of *str*

```
filename =
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/ioplugins/virtualfilesystemvaluelist.py'
```

```
module = <module 'virtualfilesystemvaluelist' from
```

```
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/
site-packages/fastr/resources/plugins/ioplugins/virtualfilesystemvaluelist.py'>
```

```
scheme = 'vfslist'
```

```
class fastr.plugins.XNATStorage
```

Bases: *IOPlugin*

**Warning:** As this `IOPlugin` is under development, it has not been thoroughly tested.

The `XNATStorage` plugin is an `IOPlugin` that can download data from and upload data to an XNAT server. It uses its own `xnat://` URL scheme. This is a scheme specific for this plugin and though it looks somewhat like the XNAT rest interface, a different type or URL.

Data resources can be access directly by a data url:

```
xnat://xnat.example.com/data/archive/projects/sandbox/subjects/subject001/  
↳experiments/experiment001/scans/T1/resources/DICOM  
xnat://xnat.example.com/data/archive/projects/sandbox/subjects/subject001/  
↳experiments/*_BRAIN/scans/T1/resources/DICOM
```

In the second URL you can see a wildcard being used. This is possible as long as it resolves to exactly one item.

The id query element will change the field from the default experiment to subject and the label query element sets the use of the label as the fastr id (instead of the XNAT id) to True (the default is False)

To disable https transport and use http instead the query string can be modified to add insecure=true. This will make the plugin send requests over http:

```
xnat://xnat.example.com/data/archive/projects/sandbox/subjects/subject001/  
↳experiments/*_BRAIN/scans/T1/resources/DICOM?insecure=true
```

For sinks it is import to know where to save the data. Sometimes you want to save data in a new assessor/resource and it needs to be created. To allow the Fastr sink to create an object in XNAT, you have to supply the type as a query parameter:

```
xnat://xnat.bmia.nl/data/archive/projects/sandbox/subjects/S01/experiments/_BRAIN/  
↳assessors/test_assessor/resources/IMAGE/files/image.nii.gz?resource_  
↳type=xnat:resourceCatalog&assessor_type=xnat:qcAssessmentData
```

Valid options are: subject\_type, experiment\_type, assessor\_type, scan\_type, and resource\_type.

If you want to do a search where multiple resources are returned, it is possible to use a search url:

```
xnat://xnat.example.com/search?projects=sandbox&subjects=subject[0-9][0-9][0-9]&  
↳experiments=*_BRAIN&scans=T1&resources=DICOM
```

This will return all DICOMs for the T1 scans for experiments that end with \_BRAIN that belong to a subjectXXX where XXX is a 3 digit number. By default the ID for the samples will be the experiment XNAT ID (e.g. XNAT\_E00123). The wildcards that can be used are the same UNIX shell-style wildcards as provided by the module [fnmatch](#).

It is possible to change the id to a different fields id or label. Valid fields are project, subject, experiment, scan, and resource:

```
xnat://xnat.example.com/search?projects=sandbox&subjects=subject[0-9][0-9][0-9]&  
↳experiments=*_BRAIN&scans=T1&resources=DICOM&id=subject&label=true
```

The following variables can be set in the search query:



variable	default	usage
projects	*	The project(s) to select, can contain wildcards (see <a href="#">fnmatch</a> )
subjects	*	The subject(s) to select, can contain wildcards (see <a href="#">fnmatch</a> )
experiments	*	The experiment(s) to select, can contain wildcards (see <a href="#">fnmatch</a> )
scans	*	The scan(s) to select, can contain wildcards (see <a href="#">fnmatch</a> )
resources	*	The resource(s) to select, can contain wildcards (see <a href="#">fnmatch</a> )
id	experiment	What field to use as the id, can be: project, subject, experiment, scan, or resource
label	false	Indicate the XNAT label should be used as fastr id, options <code>true</code> or <code>false</code>
insecure	false	Change the url scheme to be used to http instead of https
verify	true	(Dis)able the verification of SSL certificates
regex	false	Change search to use regex <a href="#">re.match()</a> instead of <a href="#">fnmatch</a> for matching
overwrite	false	Tell XNAT to overwrite existing files if a file with the name is already present

For storing credentials the `.netrc` file can be used. This is a common way to store credentials on UNIX systems. It is required that the file is only accessible by the owner only or a `NetrcParseError` will be raised. A `netrc` file is really easy to create, as its entries look like:

```
machine xnat.example.com
  login username
  password secret123
```

See the [netrc module](#) or the [GNU inet utils website](#) for more information about the `.netrc` file.

**Note:** On windows the location of the `netrc` file is assumed to be `os.path.expanduser('~/_netrc')`. The leading underscore is because windows does not like filename starting with a dot.

**Note:** For scan the label will be the scan type (this is initially the same as the series description, but can be updated manually or the XNAT scan type cleanup).

**Warning:** labels in XNAT are not guaranteed to be unique, so be careful when using them as the sample ID.

For background on XNAT, see the [XNAT API DIRECTORY](#) for the REST API of XNAT.

```
__abstractmethods__ = frozenset({})
```

```
__init__()
```

Initialization for the IOPlugin

**Returns**

newly created IOPlugin

```
__module__ = 'fastr.plugins'
```

```
cleanup()
```

(abstract) Clean up the IOPlugin. This is to do things like closing files or connections. Will be called when the plugin is no longer required.

**connect**(*server*, *path*="", *insecure*=False, *verify*=True)

**expand\_url**(*url*)

(abstract) Expand an URL. This allows a source to collect multiple samples from a single url. The URL will have a wildcard or point to something with info and multiple urls will be returned.

**Parameters**

**url** (*str*) – url to expand

**Returns**

the resulting url(s), a tuple if multiple, otherwise a str

**Return type**

*str* or *tuple* of *str*

**fetch\_url**(*inurl*, *outpath*)

Get the file(s) or values from XNAT.

**Parameters**

- **inurl** – url to the item in the data store
- **outpath** – path where to store the fetch data locally

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/ioplugins/xnatstorage.py'
```

```
module = <module 'xnatstorage' from  
'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/3.3.1/lib/python3.6/site-packages/fastr/resources/plugins/ioplugins/xnatstorage.py'>
```

**parse\_uri**(*url*)

**put\_url**(*inpath*, *outurl*)

Upload the files to the XNAT storage

**Parameters**

- **inpath** – path to the local data
- **outurl** – url to where to store the data in the external data store.

```
scheme = ('xnat', 'xnat+http', 'xnat+https')
```

**property server**

**static upload**(*resource*, *in\_path*, *location*, *retries*=3, *overwrite*=False)

**property xnat**

**fastr.plugins.json**

alias of `JsonCollector`

**fastr.plugins.path**

alias of `PathCollector`

**fastr.plugins.stdout**

alias of `StdoutCollector`

## executionplugin Module

```
class fastr.plugins.executionplugin.ExecutionPlugin(finished_callback=None,
                                                    cancelled_callback=None)
```

Bases: Plugin

This class is the base for all Plugins to execute jobs somewhere. There are many methods already in place for taking care of stuff.

There are fall-backs for certain features, but if a system already implements those it is usually preferred to skip the fall-back and let the external system handle it. There are a few flags to enable/disable these features:

- `cls.SUPPORTS_CANCEL` indicates that the plugin can cancel queued jobs
- `cls.SUPPORTS_HOLD_RELEASE` indicates that the plugin can queue jobs in a hold state and can release them again (if not, the base plugin will create a hidden queue for held jobs). The plugin should respect the `Job.status == JobState.hold` when queueing jobs.
- `cls.SUPPORTS_DEPENDENCY` indicate that the plugin can manage job dependencies, if not the base plugin job dependency system will be used and jobs will only be submitted when all dependencies are met.
- `cls.CANCELS_DEPENDENCIES` indicates that if a job is cancelled it will automatically cancel all jobs depending on that job. If not the plugin traverses the dependency graph and kills each job manually.

---

**Note:** If a plugin supports dependencies it is assumed that when a job gets cancelled, the depending job also gets cancelled automatically!

---

Most plugins should only need to redefine a few abstract methods:

- `__init__` the constructor
- `cleanup` a clean up function that frees resources, closes connections, etc
- `_queue_job` the method that queues the job for execution

Optionally an extra job finished callback could be added:

- `_job_finished` extra callback for when a job finishes

If `SUPPORTS_CANCEL` is set to `True`, the plugin should also implement:

- `_cancel_job` cancels a previously queued job

If `SUPPORTS_HOLD_RELEASE` is set to `True`, the plugin should also implement:

- `_hold_job` holds a job that is currently held
- `_release_job` releases a job that is currently held

If `SUPPORTED_DEPENDENCY` is set to `True`, the plugin should:

- Make sure to use the `Job.hold_jobs` as a list of its dependencies

Not all of the functions need to actually do anything for a plugin. There are examples of plugins that do not really need a `cleanup`, but for safety you need to implement it. Just using a `pass` for the method could be fine in such a case.

**Warning:** When overwriting other functions, extreme care must be taken not to break the plugins working, as there is a lot of bookkeeping that can go wrong.

**CANCELS\_DEPENDENCIES = False**

Indicates that when a job is cancelled the dependencies

**SUPPORTS\_CANCEL = False**

Indicates if the plugin can cancel queued jobs

**SUPPORTS\_DEPENDENCY = False**

Indicate if the plugin can manage job dependencies, if not the base plugin job dependency system will be used and jobs will only be submitted when all dependencies are met.

**SUPPORTS\_HOLD\_RELEASE = False**

Indicates if the plugin can queue jobs in a hold state and can release them again (if not, the base plugin will create a hidden queue for held jobs)

**\_\_abstractmethods\_\_ = frozenset({'\_\_init\_\_', '\_queue\_job', 'cleanup'})**

**\_\_del\_\_()**

Cleanup if the variable was deleted on purpose

**\_\_enter\_\_()**

**\_\_exit\_\_(type\_, value, tb)**

**abstract \_\_init\_\_(finished\_callback=None, cancelled\_callback=None)**

Setup the ExecutionPlugin

**Parameters**

- **finished\_callback** – the callback to call after a job finished
- **cancelled\_callback** – the callback to call after a job cancelled

**Returns**

newly created ExecutionPlugin

**\_\_module\_\_ = 'fastr.plugins.executionplugin'**

**cancel\_job(job)**

Cancel a job previously queued

**Parameters**

**job** – job to cancel

**check\_job\_requirements(job\_id)**

Check if the requirements for a job are fulfilled.

**Parameters**

**job\_id** – job to check

**Returns**

directive what should happen with the job

**Return type**

*JobAction*

**check\_job\_status(job\_id)**

Get the status of a specified job

**Parameters**

**job\_id** – the target job

**Returns**

the status of the job (or None if job not found)

**check\_nr\_queued\_jobs()**

**clean\_free\_jobs(*job*)**

**abstract cleanup()**

Method to call to clean up the ExecutionPlugin. This can be to clear temporary data, close connections, etc.

**Parameters**

**force** – force cleanup (e.g. kill instead of join a process)

**get\_job(*job\_id*)**

**get\_status(*job*)**

**hold\_job(*job*)**

**job\_finished(*job*, *errors=None*, *blocking=False*)**

The default callback that is called when a Job finishes. This will create a new thread that handles the actual callback.

**Parameters**

- **job** (*Job*) – the job that finished
- **errors** – optional list of errors encountered
- **blocking** (*bool*) – if blocking, do not create threads

**Returns**

**process\_callbacks()**

**queue\_job(*job*)**

Add a job to the execution queue

**Parameters**

**job** (*Job*) – job to add

**register\_job(*job*)**

**release\_job(*job*)**

Release a job that has been put on hold

**Parameters**

**job** – job to release

**show\_jobs(*req\_status=None*)**

List the queued jobs, possible filtered by status

**Parameters**

**req\_status** – requested status to filter on

**Returns**

list of jobs

**signal\_dependent\_jobs(*job\_id*)**

Check all dependent jobs and process them if all their dependencies are met. :param job\_id: :return:

```
class fastr.plugins.executionplugin.JobAction(value)
```

Bases: [Enum](#)

Job actions that can be performed. This is used for checking if held jobs should be queued, held longer or be cancelled.

```
__module__ = 'fastr.plugins.executionplugin'
```

```
cancel = 'cancel'
```

```
hold = 'hold'
```

```
queue = 'queue'
```

## reportingplugin Module

```
class fastr.plugins.reportingplugin.ReportingPlugin
```

Bases: [Plugin](#)

Base class for all reporting plugins. The plugin has a number of methods that can be implemented that will be called on certain events. On these events the plugin can inspect the presented data and take reporting actions.

```
__abstractmethods__ = frozenset({})
```

```
__module__ = 'fastr.plugins.reportingplugin'
```

```
activate()
```

```
deactivate()
```

```
job_updated(job)
```

```
log_record_emitted(record)
```

```
run_finished(run)
```

```
run_started(run)
```

## Subpackages

### managers Package

### managers Package

### executionpluginmanager Module

This module holds the ExecutionPluginManager as well as the base-class for all ExecutionPlugins.

```
class fastr.plugins.managers.executionpluginmanager.ExecutionPluginManager(parent)
```

Bases: [PluginSubManager](#)

Container holding all the ExecutionPlugins known to the Fastr system

```
__abstractmethods__ = frozenset({})
```

```

__args__ = None

__extra__ = None

__init__(parent)
    Initialize a ExecutionPluginManager and load plugins.

    Parameters
        • path – path to search for plugins
        • recursive – flag for searching recursively

    Returns
        newly created ExecutionPluginManager

__module__ = 'fastr.plugins.managers.executionpluginmanager'

__next_in_mro__
    alias of object

__orig_bases__ = (fastr.plugins.managers.pluginmanager.PluginSubManager,)

__origin__ = None

__parameters__ = ()

__subclasshook__()
    Abstract classes can override this to customize issubclass().

    This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__tree_hash__ = -9223366124441394047

```

### interfacemanager Module

This module holds the ExecutionPluginManager as well as the base-class for all ExecutionPlugins.

**class** `fastr.plugins.managers.interfacemanager.InterfacePluginManager`(*parent*)

Bases: `PluginSubManager`

Container holding all the CollectorPlugins

```
__abstractmethods__ = frozenset({})
```

```
__args__ = None
```

```
__extra__ = None
```

```
__init__(parent)
```

Create the Coll :param path: :param recursive: :return:

```
__module__ = 'fastr.plugins.managers.interfacemanager'
```

```
__next_in_mro__
```

alias of `object`

```
__orig_bases__ = (fastr.plugins.managers.pluginmanager.PluginSubManager,)
__origin__ = None
__parameters__ = ()
__subclasshook__()
    Abstract classes can override this to customize issubclass().

    This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).
__tree_hash__ = -9223366124441393532
```

### iopluginmanager Module

```
class fastr.plugins.managers.iopluginmanager.IOPluginManager(parent)
    Bases: PluginSubManager
    A mapping containing the IOPlugins known to this system
    __abstractmethods__ = frozenset({})
    __args__ = None
    __extra__ = None
    __init__(parent)
        Create the IOPluginManager and populate it.
        Returns
            newly created IOPluginManager
    __iter__()
        Get an iterator from the BaseManager. The iterator will iterate over the keys of the BaseManager.
        Returns
            the iterator
        Return type
            dictionary-keyiterator
    __keytransform__(key)
        Identity transform for the keys. This function can be reimplemented by a subclass to implement a different key transform.
        Parameters
            key – key to transform
        Returns
            the transformed key (in this case the same key as inputted)
    __module__ = 'fastr.plugins.managers.iopluginmanager'
    __next_in_mro__
        alias of object
    __orig_bases__ = (fastr.plugins.managers.pluginmanager.PluginSubManager,)
```



**\_\_origin\_\_** = None

**\_\_parameters\_\_** = ()

**\_\_subclasshook\_\_**()

Abstract classes can override this to customize `issubclass()`.

This is invoked early on by `abc.ABCMeta.__subclasscheck__()`. It should return `True`, `False` or `NotImplemented`. If it returns `NotImplemented`, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

**\_\_tree\_hash\_\_** = -9223366124441393126

**cleanup**()

Cleanup all plugins, this closes files, connections and other things that could be left dangling otherwise.

**static create\_ioplugintool**(*tools, interfaces*)

Create the tools which handles sinks and sources. The command of this tool is the main of `core.ioplugin`.

**expand\_url**(*url*)

Expand the url by filling the wildcards. This function checks the url scheme and uses the `expand` function of the correct IOPlugin.

**Parameters**

**url** (*str*) – url to expand

**Returns**

list of urls

**Return type**

list of *str*

**pull\_source\_data**(*url, outdir, sample\_id, datatype=None*)

Retrieve data from an external source. This function checks the url scheme and selects the correct IOPlugin to retrieve the data.

**Parameters**

- **url** – url to pull
- **outdir** (*str*) – the directory to write the data to
- **datatype** (*DataType*) – the datatype of the data, used for determining the total contents of the transfer

**Returns**

None

**push\_sink\_data**(*inpath, outurl, datatype=None*)

Send data to an external source. This function checks the url scheme and selects the correct IOPlugin to retrieve the data.

**Parameters**

- **inpath** (*str*) – the path of the data to be pushed
- **outurl** (*str*) – the url to write the data to
- **datatype** (*DataType*) – the datatype of the data, used for determining the total contents of the transfer

**put\_url**(*inpath*, *outurl*)

Put the files to the external data store.

**Parameters**

- **inpath** – path to the local data
- **outurl** – url to where to store the data in the external data store.

**static register\_url\_scheme**(*scheme*)

Register a custom scheme to behave http like. This is needed to parse all things properly with urlparse.

**Parameters**

**scheme** – the scheme to register

**url\_to\_path**(*url*)

Retrieve the path for a given url

**Parameters**

**url** (*str*) – the url to parse

**Returns**

the path corresponding to the input url

**Return type**

*str*

## networkmanager Module

This module contains the tool manager class

**class** fastr.plugins.managers.networkmanager.**NetworkManager**(*path*)

Bases: *ObjectManager*

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_args\_\_** = None

**\_\_extra\_\_** = None

**\_\_module\_\_** = 'fastr.plugins.managers.networkmanager'

**\_\_next\_in\_mro\_\_**

alias of *object*

**\_\_orig\_bases\_\_** = (fastr.plugins.managers.objectmanager.ObjectManager,)

**\_\_origin\_\_** = None

**\_\_parameters\_\_** = ()

**\_\_subclasshook\_\_**()

Abstract classes can override this to customize *issubclass*().

This is invoked early on by *abc.ABCMeta.\_\_subclasscheck\_\_*(). It should return *True*, *False* or *NotImplemented*. If it returns *NotImplemented*, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

**\_\_tree\_hash\_\_** = -9223366124441392264

**get\_object\_version(obj)**

Get the version of a given object

**Parameters**

**object** – the object to use

**Returns**

the version of the object

**property object\_class**

The class of the objects to populate the manager with

**objectmanager Module**

This module contains the object manager class

**class** fastr.plugins.managers.objectmanager.**ObjectManager**(path)

Bases: BaseManager

Class for managing all the objects loaded in the fastr system

**\_\_abstractmethods\_\_** = frozenset({'get\_object\_version', 'object\_class'})

**\_\_args\_\_** = None

**\_\_contains\_\_**(key)

Check if an item is in the ObjectManager

**Parameters**

**key** (*str* or *tuple*) – object id or tuple (Objectid, version)

**Returns**

flag indicating the item is in the manager

**\_\_extra\_\_** = None

**\_\_getitem\_\_**(key)

Retrieve a Object from the ObjectManager. You can request by only an id, which results in the newest version of the Object being returned, or request using both an id and a version.

**Parameters**

**key** (*str* or *tuple*) – object id or tuple (Objectid, version)

**Returns**

the requested Object

**Raises**

**FastrObjectUnknownError** – if a non-existing Object was requested

**\_\_init\_\_**(path)

Create a ObjectManager and scan path to search for Objects

**Parameters**

**path** (*str* or *iterable of str*) – the path(s) to scan for Objects

**Returns**

newly created ObjectManager

**\_\_keytransform\_\_**(*key*)

Key transform, used for allowing indexing both by id-only and by (id, version)

**Parameters**

**key** – key to transform

**Returns**

key in form (id, version)

**\_\_module\_\_** = 'fastr.plugins.managers.objectmanager'

**\_\_next\_in\_mro\_\_**

alias of `object`

**\_\_orig\_bases\_\_** = (fastr.abc.basemanager.BaseManager,)

**\_\_origin\_\_** = None

**\_\_parameters\_\_** = ()

**\_\_subclasshook\_\_**()

Abstract classes can override this to customize `issubclass()`.

This is invoked early on by `abc.ABCMeta.__subclasscheck__()`. It should return `True`, `False` or `NotImplemented`. If it returns `NotImplemented`, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

**\_\_tree\_hash\_\_** = -9223366124441392901

**abstract get\_object\_version**(*obj*)

Get the version of a given object

**Parameters**

**object** – the object to use

**Returns**

the version of the object

**abstract property object\_class**

The class of the objects to populate the manager with

**objectversions**(*obj*)

Return a list of available versions for the object

**Parameters**

**object** – The object to check the versions for. Can be either a *Object* or a *str*.

**Returns**

List of version objects. Returns *None* when the given object is not known.

**todict**()

Return a dictionary version of the Manager

**Returns**

manager as a dict

## pluginmanager Module

This module contains the Manager class for Plugins in the fastr system

**class** fastr.plugins.managers.pluginmanager.**PluginManager**(*path=None*)

Bases: BasePluginManager

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_args\_\_** = None

**\_\_extra\_\_** = None

**\_\_init\_\_**(*path=None*)

Create a BasePluginManager and scan the give path for matching plugins

### Parameters

- **path** (*str*) – path to scan
- **recursive** (*bool*) – flag to indicate a recursive search
- **module** (*module*) – the module to register plugins into

### Returns

newly created plugin manager

### Raises

**FastrTypeError** – if self.\_plugin\_class is set to a class not subclassing BasePlugin

**\_\_module\_\_** = 'fastr.plugins.managers.pluginmanager'

**\_\_next\_in\_mro\_\_**

alias of *object*

**\_\_orig\_bases\_\_** = (fastr.abc.basepluginmanager.BasePluginManager,)

**\_\_origin\_\_** = None

**\_\_parameters\_\_** = ()

**\_\_setitem\_\_**(*key, value*)

Store an item in the BaseManager, will ignore the item if the key is already present in the BaseManager.

### Parameters

- **name** – the key of the item to save
- **value** – the value of the item to save

### Returns

None

**\_\_subclasshook\_\_**()

Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.\_\_subclasscheck\_\_(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

**\_\_tree\_hash\_\_** = -9223366124441394677

**property plugin\_class**

The plugin manager contains any Plugin subclass

```
class fastr.plugins.managers.pluginmanager.PluginSubManager(parent, plugin_class)
```

Bases: BasePluginManager

A PluginManager that is a selection of a parent plugin manger. It uses the PluginsView to only expose part of the parent PluginManager. This is used to create plugin plugins.managers for only certain types of plugins (e.g. IOPlugins) without loading them multiple times.

```
__abstractmethods__ = frozenset({})
```

```
__args__ = None
```

```
__extra__ = None
```

```
__init__(parent, plugin_class)
```

Create a BasePluginManager and scan the give path for matching plugins

**Parameters**

- **path** (*str*) – path to scan
- **recursive** (*bool*) – flag to indicate a recursive search
- **module** (*module*) – the module to register plugins into

**Returns**

newly created plugin manager

**Raises**

**FastrTypeError** – if self.\_plugin\_class is set to a class not subclassing BasePlugin

```
__module__ = 'fastr.plugins.managers.pluginmanager'
```

```
__next_in_mro__
```

alias of *object*

```
__orig_bases__ = (fastr.abc.basepluginmanager.BasePluginManager,)
```

```
__origin__ = None
```

```
__parameters__ = ()
```

```
__subclasshook__()
```

Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.\_\_subclasscheck\_\_(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

```
__tree_hash__ = -9223366124441394736
```

**property data**

The actual data dict underlying this Manager

**property plugin\_class**

PluginSubManagers only expose the plugins of a certain class

```
class fastr.plugins.managers.pluginmanager.PluginsView(parent, plugin_class)
```

Bases: `MutableMapping`

A collection that acts like view of the plugins of another plugin manager. This is a proxy object that only gives access the plugins of a certain plugin class. It behaves like a mapping and is used as the data object for a `PluginSubManager`.

```
__abstractmethods__ = frozenset({})
```

```
__delitem__(key)
```

```
__dict__ = mappingproxy({'__module__': 'fastr.plugins.managers.pluginmanager',
'__doc__': '\n A collection that acts like view of the plugins of another plugin
manager.\n This is a proxy object that only gives access the plugins of a certain\n
plugin class. It behaves like a mapping and is used as the data object for\n a
PluginSubManager.\n ', '__init__': <function PluginsView.__init__>,
'filter_plugin': <function PluginsView.filter_plugin>, '__getitem__': <function
PluginsView.__getitem__>, '__setitem__': <function PluginsView.__setitem__>,
'__delitem__': <function PluginsView.__delitem__>, '__len__': <function
PluginsView.__len__>, '__iter__': <function PluginsView.__iter__>, '__dict__':
<attribute '__dict__' of 'PluginsView' objects>, '__weakref__': <attribute
'__weakref__' of 'PluginsView' objects>, '__abstractmethods__': frozenset(),
'_abc_registry': <_weakrefset.WeakSet object>, '_abc_cache': <_weakrefset.WeakSet
object>, '_abc_negative_cache': <_weakrefset.WeakSet object>,
'_abc_negative_cache_version': 59, '__annotations__': {}})
```

```
__getitem__(item)
```

```
__init__(parent, plugin_class)
```

Constructor for the plugins view

#### Parameters

- **parent** (`BasePluginManager`) – the parent plugin manager
- **plugin\_class** (`class`) – the class of the plugins to expose

```
__iter__()
```

```
__len__()
```

```
__module__ = 'fastr.plugins.managers.pluginmanager'
```

```
__setitem__(key, value)
```

```
__weakref__
```

list of weak references to the object (if defined)

```
filter_plugin(plugin)
```

## targetmanager Module

This module holds the ExecutionPluginManager as well as the base-class for all ExecutionPlugins.

**class** fastr.plugins.managers.targetmanager.TargetManager(*parent*)

Bases: *PluginSubManager*

Container holding all the ExecutionPlugins known to the Fastr system

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_args\_\_** = None

**\_\_extra\_\_** = None

**\_\_init\_\_**(*parent*)

Initialize a ExecutionPluginManager and load plugins.

**Returns**

newly created ExecutionPluginManager

**\_\_module\_\_** = 'fastr.plugins.managers.targetmanager'

**\_\_next\_in\_mro\_\_**

alias of *object*

**\_\_orig\_bases\_\_** = (fastr.plugins.managers.pluginmanager.PluginSubManager,)

**\_\_origin\_\_** = None

**\_\_parameters\_\_** = ()

**\_\_subclasshook\_\_**()

Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.\_\_subclasscheck\_\_(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

**\_\_tree\_hash\_\_** = -9223366124441392997

## toolmanager Module

This module contains the tool manager class

**class** fastr.plugins.managers.toolmanager.ToolManager(*path*)

Bases: *ObjectManager*

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_args\_\_** = None

**\_\_extra\_\_** = None

**\_\_module\_\_** = 'fastr.plugins.managers.toolmanager'

**\_\_next\_in\_mro\_\_**

alias of *object*



```
__orig_bases__ = (fastr.plugins.managers.objectmanager.ObjectManager,)
```

```
__origin__ = None
```

```
__parameters__ = ()
```

```
__subclasshook__()
```

Abstract classes can override this to customize `issubclass()`.

This is invoked early on by `abc.ABCMeta.__subclasscheck__()`. It should return `True`, `False` or `NotImplemented`. If it returns `NotImplemented`, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

```
__tree_hash__ = -9223366124441392323
```

```
get_object_version(obj)
```

Get the version of a given object

**Parameters**

**object** – the object to use

**Returns**

the version of the object

```
property object_class
```

The class of the objects to populate the manager with

```
populate()
```

Populate the manager with the data. This is a method that will be called when the Managers data is first accessed. This way we avoid doing expensive directory scans when the data is never requested.

```
toolversions(tool)
```

Return a list of available versions for the tool

**Parameters**

**tool** – The tool to check the versions for. Can be either a *Tool* or a *str*.

**Returns**

List of version objects. Returns *None* when the given tool is not known.

**test Package**

**test Package**

**test\_datatypes Module**

**utils Package**

**utils Package**

A collections of utils for fastr (command line tools or non-core functionality)

## compare Module

Module to compare various fastr specific things such as a execution directory or a reference directory.

`fastr.utils.compare.compare_execution_dir(path1, path2)`

`fastr.utils.compare.compare_job_dirs(sample, node, node_dir1, node_dir2)`

`fastr.utils.compare.compare_job_output_data(output, job1, job2)`

`fastr.utils.compare.compare_set(set1, set2, path, sub_compare_func, f_args=None, f_kwargs=None)`

Compare two sets and dispatch each item to a sub comparison function

### Parameters

- **set1** (*Iterable*) – first set of items
- **set2** (*Iterable*) – second set of items
- **path** (*str*) – identifier of the data location
- **sub\_compare\_func** – function to apply to items
- **f\_args** – args to pass to sub\_compare\_func
- **f\_kwargs** – kwargs to pass to sub\_compare\_func

### Returns

generator that iterates over the differences

### Return type

generator

`fastr.utils.compare.compare_value_dict_item(key, data1, data2, path)`

`fastr.utils.compare.compare_value_list(data1, data2, path, key=None)`

## dicteq Module

Some helper function to compare dictionaries and find the parts of the dict that are different. This is mostly to help in debugging.

`fastr.utils.dicteq.dicteq(self, other)`

Compare two dicts for equality

### Parameters

- **self** – the first object to compare
- **other** – the oth

### Returns

`fastr.utils.dicteq.diffdict(self, other, path=None, visited=None)`

Find the differences in two dictionaries.

### Parameters

- **self** – the first object to compare
- **other** (*dict*) – other dictionary
- **path** (*list*) – the path for nested dicts (too keep track of recursion)

**Returns**

list of messages indicating the differences

**Return type**

`list`

`fastr.utils.dicteq.diffobj(self, other, path=None, visited=None)`

Compare two objects by comparing their `__dict__` entries

**Parameters**

- **self** – the first object to compare
- **other** – other objects to compare
- **path** (`list`) – the path for nested dicts (too keep track of recursion)

**Returns**

list of messages

**Return type**

`list`

`fastr.utils.dicteq.diffobj_str(self, other)`

Compare two objects by comparing their `__dict__` entries, but returns the differences in a single string ready for logging.

**Parameters**

- **self** – the first object to compare
- **other** – other object to compare to

**Returns**

the description of the differences

**Return type**

`str`

## gettools Module

`fastr.utils.gettools.main()`

## multiprocesswrapper Module

`fastr.utils.multiprocesswrapper.function_wrapper(filepath, fnc_name, *args, **kwargs)`

## verify Module

`fastr.utils.verify.create_tool_test(filename, log=<Logger fastr (INFO)>)`

Create test for fastr verify tool.

By running `fastr verify -c tool FILENAME` the input data in the folders under ‘tests’ in the tool definition is processed by the tool. The output data is written to a folder in each test folder. In each test folder a gzipped pickle is created which is used to verify the working of the tool at a later time.

**Parameters**

- **filename** – filename of the tool definition
- **log** – the logger to use to send messages to

`fastr.utils.verify.verify_resource_loading(filename, log=<Logger fastr (INFO)>)`

Verify that a resource file can be loaded. Returns loaded object.

**Parameters**

- **filename** (`str`) – path of the object to load
- **log** – the logger to use to send messages to

**Returns**

loaded resource

`fastr.utils.verify.verify_tool(filename, log=<Logger fastr (INFO)>, perform_tests=True)`

Verify that a tool correctly works. Returns Tool.

**Parameters**

- **filename** – filename of the tool definition
- **log** – the logger to use to send messages to
- **perform\_test** – Boolean to

**Returns**

Tool object

`fastr.utils.verify.verify_tool_instantiate(doc, filename, log=<Logger fastr (INFO)>)`

Verify the tool schema. Returns checked loaded object.

**Parameters**

- **doc** – loaded object
- **filename** – filename of the tool definition
- **log** – the logger to use to send messages to

**Returns**

Tool object

`fastr.utils.verify.verify_tool_schema(doc, log=<Logger fastr (INFO)>)`

Verify the tool schema. Returns checked loaded object.

**Parameters**

- **doc** – loaded object to check
- **log** – the logger to use to send messages to

**Returns**

object with checked schema

## Subpackages

### cmd Package

#### cmd Package

`fastr.utils.cmd.add_parser_doc_link(parser, filepath)`

`fastr.utils.cmd.find_commands()`

`fastr.utils.cmd.get_command_module(command)`

`fastr.utils.cmd.main()`

`fastr.utils.cmd.print_help(commands=None)`

#### cat Module

`fastr.utils.cmd.cat.fastr_cat(infile, path)`

`fastr.utils.cmd.cat.get_parser()`

`fastr.utils.cmd.cat.main()`

Print information from a job file

#### dump Module

`fastr.utils.cmd.dump.create_zip(directory, output_file)`

`fastr.utils.cmd.dump.get_parser()`

`fastr.utils.cmd.dump.main()`

Dump the contents of a network run tempdir into a zip for remote assistance

#### execute Module

`fastr.utils.cmd.execute.get_parser()`

`fastr.utils.cmd.execute.main()`

Execute a fastr job file

#### extract\_argparse Module

`fastr.utils.cmd.extract_argparse.cardinality_from_nargs(value)`

`fastr.utils.cmd.extract_argparse.datatype_from_type(type_, metavar)`

`fastr.utils.cmd.extract_argparse.extract_argparser(filepath)`

```
fastr.utils.cmd.extract_argparse.find_argparser(entry,  
                                                  basename='/home/docs/checkouts/readthedocs.org/user_builds/fastr/env  
packages/sphinx/__main__.py')
```

```
fastr.utils.cmd.extract_argparse.get_parser()
```

```
fastr.utils.cmd.extract_argparse.main()
```

Create a stub for a Tool based on a python script using argparse

### **provenance Module**

```
fastr.utils.cmd.provenance.get_parser()
```

```
fastr.utils.cmd.provenance.get_prov_document(result)
```

```
fastr.utils.cmd.provenance.main()
```

Get PROV information from the result pickle.

### **pylint Module**

```
fastr.utils.cmd.pylint.get_parser()
```

```
fastr.utils.cmd.pylint.main()
```

Tiny wrapper in pylint so the output can be saved to a file (for test automation)

```
fastr.utils.cmd.pylint.run_pylint(out_file, pylint_args)
```

### **report Module**

```
fastr.utils.cmd.report.get_parser()
```

```
fastr.utils.cmd.report.main()
```

Print report of a job result (\_\_fastr\_result\_\_.pickle.gz) file

### **run Module**

```
fastr.utils.cmd.run.create_network_parser(network)
```

```
fastr.utils.cmd.run.get_parser()
```

```
fastr.utils.cmd.run.main()
```

Run a Network from the commandline

### sink Module

```
fastr.utils.cmd.sink.get_parser()
fastr.utils.cmd.sink.main()
    Command line access to the IOPlugin sink
fastr.utils.cmd.sink.sink()
```

### source Module

```
fastr.utils.cmd.source.get_parser()
fastr.utils.cmd.source.main()
    Command line access to the IOPlugin source
fastr.utils.cmd.source.source()
```

### test Module

```
fastr.utils.cmd.test.check_network(args)
fastr.utils.cmd.test.check_networks(args)
fastr.utils.cmd.test.check_tool(args)
fastr.utils.cmd.test.check_tools(args)
fastr.utils.cmd.test.directory(path)
    Make sure the path is a valid directory for argparse
fastr.utils.cmd.test.get_parser()
fastr.utils.cmd.test.main()
    Run the tests of a tool to verify the proper function
fastr.utils.cmd.test.tool(value)
    Make sure the value is a correct tool for argparse or reference directory
```

### trace Module

```
fastr.utils.cmd.trace.get_parser()
fastr.utils.cmd.trace.main()
    Trace samples/sinks from a run
fastr.utils.cmd.trace.print_sample_sink(sink_data, dirname, sample_sink_tuples, verbose)
fastr.utils.cmd.trace.print_samples(sink_data, sample_ids, verbose)
fastr.utils.cmd.trace.print_sinks(sink_data, sink_ids, verbose)
fastr.utils.cmd.trace.read_sink_data(infile)
fastr.utils.cmd.trace.switch_sample_sink(sink_data)
```

## upgrade Module

```
class fastr.utils.cmd.upgrade.FastrNamespaceType(toollist, typelist)
    Bases: tuple
    __getnewargs__()
        Return self as a plain tuple. Used by copy and pickle.
    __module__ = 'fastr.utils.cmd.upgrade'
    static __new__(cls, toollist, typelist)
        Create new instance of FastrNamespaceType(toollist, typelist)
    __repr__()
        Return a nicely formatted representation string
    __slots__ = ()
    property toollist
        Alias for field number 0
    property typelist
        Alias for field number 1

class fastr.utils.cmd.upgrade.dummy_container
    Bases: object
    __dict__ = mappingproxy({'__module__': 'fastr.utils.cmd.upgrade', '__getitem__':
    <function dummy_container.__getitem__>, '__dict__': <attribute '__dict__' of
    'dummy_container' objects>, '__weakref__': <attribute '__weakref__' of
    'dummy_container' objects>, '__doc__': None, '__annotations__': {}})
    __getitem__(value)
    __module__ = 'fastr.utils.cmd.upgrade'
    __weakref__
        list of weak references to the object (if defined)

fastr.utils.cmd.upgrade.find_tool(toolspec)

fastr.utils.cmd.upgrade.get_parser()

fastr.utils.cmd.upgrade.main()
    Upgrade a fastr 2.x python file to fastr 3.x syntax

fastr.utils.cmd.upgrade.upgrade_network(infile, outfile)

fastr.utils.cmd.upgrade.upgrade_tool(infile, outfile)
```



## **verify Module**

`fastr.utils.cmd.verify.get_parser()`

`fastr.utils.cmd.verify.main()`

Verify fastr resources, at the moment only tool definitions are supported.

## **secrets Package**

### **secrets Package**

### **secretprovider Module**

### **secretservice Module**

## **Subpackages**

### **exceptions Package**

### **exceptions Package**

### **couldnotdeletecredentials Module**

### **couldnotretrievecredentials Module**

### **couldnotsetcredentials Module**

### **notimplemented Module**

### **providernotfound Module**

### **providers Package**

### **providers Package**

### **keyringprovider Module**

### **netrcprovider Module**



## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### f

- `fastr.__init__`, 85
- `fastr.api`, 95
- `fastr.core`, 97
  - `fastr.core.cardinality`, 97
  - `fastr.core.dimension`, 104
  - `fastr.core.interface`, 106
  - `fastr.core.ioplugin`, 108
  - `fastr.core.provenance`, 111
  - `fastr.core.resourcelimit`, 112
  - `fastr.core.samples`, 114
  - `fastr.core.target`, 125
  - `fastr.core.test`, 136
  - `fastr.core.tool`, 128
  - `fastr.core.version`, 131
  - `fastr.core.vfs`, 133
- `fastr.data`, 137
  - `fastr.data.url`, 137
- `fastr.datatypes`, 139
- `fastr.exceptions`, 86
- `fastr.execution`, 151
  - `fastr.execution.basenoderun`, 151
  - `fastr.execution.environmentmodules`, 152
  - `fastr.execution.executionscript`, 154
  - `fastr.execution.flownoderun`, 154
  - `fastr.execution.inputoutputrun`, 155
  - `fastr.execution.job`, 168
  - `fastr.execution.linkrun`, 179
  - `fastr.execution.macronoderun`, 182
  - `fastr.execution.networkanalyzer`, 183
  - `fastr.execution.networkchunker`, 184
  - `fastr.execution.networkrun`, 185
  - `fastr.execution.noderun`, 187
  - `fastr.execution.sinknoderun`, 190
  - `fastr.execution.sourcenoderun`, 192
- `fastr.globals`, 94
- `fastr.helpers`, 195
  - `fastr.helpers.checksum`, 197
  - `fastr.helpers.classproperty`, 198
  - `fastr.helpers.clear_pycs`, 199
  - `fastr.helpers.configmanager`, 199
  - `fastr.helpers.events`, 206
  - `fastr.helpers.filesynchelper`, 207
  - `fastr.helpers.iohelpers`, 208
  - `fastr.helpers.jsonschemaparser`, 208
  - `fastr.helpers.lazy_module`, 210
  - `fastr.helpers.lockfile`, 211
  - `fastr.helpers.procutils`, 212
  - `fastr.helpers.report`, 212
  - `fastr.helpers.rest_generation`, 212
  - `fastr.helpers.schematotable`, 212
  - `fastr.helpers.shellescape`, 213
  - `fastr.helpers.sysinfo`, 214
  - `fastr.helpers.xmltodict`, 215
- `fastr.planning`, 216
  - `fastr.planning.inputgroup`, 216
  - `fastr.planning.inputgroupcombiner`, 218
  - `fastr.planning.inputoutput`, 220
  - `fastr.planning.link`, 237
  - `fastr.planning.network`, 240
  - `fastr.planning.node`, 244
- `fastr.plugins`, 253
  - `fastr.plugins.executionplugin`, 287
  - `fastr.plugins.managers`, 290
    - `fastr.plugins.managers.executionpluginmanager`, 290
    - `fastr.plugins.managers.interfacemanager`, 291
    - `fastr.plugins.managers.iopluginmanager`, 292
    - `fastr.plugins.managers.networkmanager`, 294
    - `fastr.plugins.managers.objectmanager`, 295
    - `fastr.plugins.managers.pluginmanager`, 297
    - `fastr.plugins.managers.targetmanager`, 300
    - `fastr.plugins.managers.toolmanager`, 300
    - `fastr.plugins.reportingplugin`, 290
- `fastr.test`, 301
- `fastr.utils`, 301
  - `fastr.utils.cmd`, 305
    - `fastr.utils.cmd.cat`, 305
    - `fastr.utils.cmd.dump`, 305
    - `fastr.utils.cmd.execute`, 305
    - `fastr.utils.cmd.extract_argparse`, 305
    - `fastr.utils.cmd.provenance`, 306
    - `fastr.utils.cmd.pylint`, 306
    - `fastr.utils.cmd.report`, 306

`fastr.utils.cmd.run`, [306](#)  
`fastr.utils.cmd.sink`, [307](#)  
`fastr.utils.cmd.source`, [307](#)  
`fastr.utils.cmd.test`, [307](#)  
`fastr.utils.cmd.trace`, [307](#)  
`fastr.utils.cmd.upgrade`, [308](#)  
`fastr.utils.cmd.verify`, [309](#)  
`fastr.utils.compare`, [302](#)  
`fastr.utils.dicteq`, [302](#)  
`fastr.utils.gettools`, [303](#)  
`fastr.utils.multiprocesswrapper`, [303](#)  
`fastr.utils.verify`, [303](#)  
`fastr.version`, [94](#)

## Symbols

- `__abstractmethods__` (*fastr.core.dimension.ForwardsDimensions attribute*), 105
- `__abstractmethods__` (*fastr.core.dimension.HasDimensions attribute*), 105
- `__abstractmethods__` (*fastr.core.interface.Interface attribute*), 107
- `__abstractmethods__` (*fastr.core.ioplugin.IOPlugin attribute*), 108
- `__abstractmethods__` (*fastr.core.samples.ContainsSamples attribute*), 114
- `__abstractmethods__` (*fastr.core.samples.HasSamples attribute*), 114
- `__abstractmethods__` (*fastr.core.samples.SampleBaseId attribute*), 115
- `__abstractmethods__` (*fastr.core.samples.SampleCollection attribute*), 116
- `__abstractmethods__` (*fastr.core.samples.SampleId attribute*), 119
- `__abstractmethods__` (*fastr.core.samples.SampleIndex attribute*), 119
- `__abstractmethods__` (*fastr.core.samples.SampleValue attribute*), 124
- `__abstractmethods__` (*fastr.core.target.ProcessUsageCollection attribute*), 125
- `__abstractmethods__` (*fastr.core.target.SubprocessBasedTarget attribute*), 126
- `__abstractmethods__` (*fastr.core.target.Target attribute*), 127
- `__abstractmethods__` (*fastr.datatypes.AnyFile attribute*), 139
- `__abstractmethods__` (*fastr.datatypes.AnyType attribute*), 140
- `__abstractmethods__` (*fastr.datatypes.BaseDataType attribute*), 140
- `__abstractmethods__` (*fastr.datatypes.DataType attribute*), 143
- `__abstractmethods__` (*fastr.datatypes.DataTypeManager attribute*), 143
- `__abstractmethods__` (*fastr.datatypes.Deferred attribute*), 147
- `__abstractmethods__` (*fastr.datatypes.EnumType attribute*), 148
- `__abstractmethods__` (*fastr.datatypes.Missing attribute*), 148
- `__abstractmethods__` (*fastr.datatypes.TypeGroup attribute*), 149
- `__abstractmethods__` (*fastr.datatypes.URLType attribute*), 149
- `__abstractmethods__` (*fastr.datatypes.ValueType attribute*), 151
- `__abstractmethods__` (*fastr.execution.basenoderun.BaseNodeRun attribute*), 152
- `__abstractmethods__` (*fastr.execution.flownoderun.AdvancedFlowNodeRun attribute*), 154
- `__abstractmethods__` (*fastr.execution.flownoderun.FlowNodeRun attribute*), 155
- `__abstractmethods__` (*fastr.execution.inputoutputrun.AdvancedFlowOutputRun attribute*), 155
- `__abstractmethods__` (*fastr.execution.inputoutputrun.BaseInputRun attribute*), 156
- `__abstractmethods__` (*fastr.execution.inputoutputrun.InputRun attribute*), 156
- `__abstractmethods__` (*fastr.execution.inputoutputrun.MacroOutputRun attribute*), 159
- `__abstractmethods__` (*fastr.execution.inputoutputrun.NamedSubinputRun attribute*), 159

*attribute*), 159

\_\_abstractmethods\_\_  
(*fastr.execution.inputoutputrun.OutputRun* *attribute*), 160

\_\_abstractmethods\_\_  
(*fastr.execution.inputoutputrun.SourceOutputRun* *attribute*), 162

\_\_abstractmethods\_\_  
(*fastr.execution.inputoutputrun.SubInputRun* *attribute*), 164

\_\_abstractmethods\_\_  
(*fastr.execution.inputoutputrun.SubOutputRun* *attribute*), 166

\_\_abstractmethods\_\_  
(*fastr.execution.linkrun.LinkRun* *attribute*), 179

\_\_abstractmethods\_\_  
(*fastr.execution.macronoderun.MacroNodeRun* *attribute*), 182

\_\_abstractmethods\_\_  
(*fastr.execution.noderun.NodeRun* *attribute*), 187

\_\_abstractmethods\_\_  
(*fastr.execution.sinknoderun.SinkNodeRun* *attribute*), 190

\_\_abstractmethods\_\_  
(*fastr.execution.sourcenoderun.ConstantNodeRun* *attribute*), 192

\_\_abstractmethods\_\_  
(*fastr.execution.sourcenoderun.SourceNodeRun* *attribute*), 193

\_\_abstractmethods\_\_  
(*fastr.planning.inputgroup.InputGroup* *attribute*), 216

\_\_abstractmethods\_\_  
(*fastr.planning.inputgroupcombiner.BaseInputGroupCombiner* *attribute*), 218

\_\_abstractmethods\_\_  
(*fastr.planning.inputgroupcombiner.DefaultInputGroupCombiner* *attribute*), 219

\_\_abstractmethods\_\_  
(*fastr.planning.inputgroupcombiner.MergingInputGroupCombiner* *attribute*), 220

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.AdvancedFlowOutput* *attribute*), 221

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.BaseInput* *attribute*), 221

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.BaseInputOutput* *attribute*), 222

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.BaseOutput* *attribute*), 224

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.Input* *attribute*), 225

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.MacroInput* *attribute*), 228

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.MacroOutput* *attribute*), 228

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.NamedSubInput* *attribute*), 228

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.Output* *attribute*), 229

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.SourceOutput* *attribute*), 231

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.SubInput* *attribute*), 232

\_\_abstractmethods\_\_  
(*fastr.planning.inputoutput.SubOutput* *attribute*), 235

\_\_abstractmethods\_\_  
(*fastr.planning.link.Link* *attribute*), 237

\_\_abstractmethods\_\_  
(*fastr.planning.node.AdvancedFlowNode* *attribute*), 244

\_\_abstractmethods\_\_  
(*fastr.planning.node.BaseNode* *attribute*), 245

\_\_abstractmethods\_\_  
(*fastr.planning.node.ConstantNode* *attribute*), 245

\_\_abstractmethods\_\_  
(*fastr.planning.node.FlowNode* *attribute*), 246

\_\_abstractmethods\_\_  
(*fastr.planning.node.MacroNode* *attribute*), 247

\_\_abstractmethods\_\_  
(*fastr.planning.node.Node* *attribute*), 248

\_\_abstractmethods\_\_  
(*fastr.planning.node.SinkNode* *attribute*), 251

\_\_abstractmethods\_\_  
(*fastr.planning.node.SourceNode* *attribute*), 252

\_\_abstractmethods\_\_  
(*fastr.plugins.BlockingExecution* *attribute*), 253

\_\_abstractmethods\_\_  
(*fastr.plugins.CommaSeperatedValueFile* *attribute*), 255

\_\_abstractmethods\_\_  
(*fastr.plugins.CrossValidation* *attribute*), 255



- attribute*), 255
- `__abstractmethods__` (*fastr.plugins.DRMAAExecution attribute*), 256
- `__abstractmethods__` (*fastr.plugins.DockerTarget attribute*), 258
- `__abstractmethods__` (*fastr.plugins.ElasticsearchReporter attribute*), 259
- `__abstractmethods__` (*fastr.plugins.FastrInterface attribute*), 261
- `__abstractmethods__` (*fastr.plugins.FileSystem attribute*), 263
- `__abstractmethods__` (*fastr.plugins.FlowInterface attribute*), 265
- `__abstractmethods__` (*fastr.plugins.HTTPPlugin attribute*), 266
- `__abstractmethods__` (*fastr.plugins.LinearExecution attribute*), 267
- `__abstractmethods__` (*fastr.plugins.LocalBinaryTarget attribute*), 269
- `__abstractmethods__` (*fastr.plugins.MacroTarget attribute*), 270
- `__abstractmethods__` (*fastr.plugins.NetworkScope attribute*), 270
- `__abstractmethods__` (*fastr.plugins.NipypeInterface attribute*), 271
- `__abstractmethods__` (*fastr.plugins.Null attribute*), 272
- `__abstractmethods__` (*fastr.plugins.PimReporter attribute*), 273
- `__abstractmethods__` (*fastr.plugins.ProcessPoolExecution attribute*), 274
- `__abstractmethods__` (*fastr.plugins.RQExecution attribute*), 275
- `__abstractmethods__` (*fastr.plugins.Reference attribute*), 276
- `__abstractmethods__` (*fastr.plugins.S3Filesystem attribute*), 276
- `__abstractmethods__` (*fastr.plugins.SimpleReport attribute*), 277
- `__abstractmethods__` (*fastr.plugins.SingularityTarget attribute*), 278
- `__abstractmethods__` (*fastr.plugins.SlurmExecution attribute*), 279
- `__abstractmethods__` (*fastr.plugins.StrongrExecution attribute*), 280
- `__abstractmethods__` (*fastr.plugins.VirtualFileSystem attribute*), 281
- `__abstractmethods__` (*fastr.plugins.VirtualFileSystemRegularExpression attribute*), 282
- `__abstractmethods__` (*fastr.plugins.VirtualFileSystemValueList attribute*), 283
- `__abstractmethods__` (*fastr.plugins.XNATStorage attribute*), 285
- `__abstractmethods__` (*fastr.plugins.executionplugin.ExecutionPlugin attribute*), 288
- `__abstractmethods__` (*fastr.plugins.managers.executionpluginmanager.ExecutionPlugin attribute*), 290
- `__abstractmethods__` (*fastr.plugins.managers.interfacemanager.InterfacePluginManager attribute*), 291
- `__abstractmethods__` (*fastr.plugins.managers.iopluginmanager.IOPluginManager attribute*), 292
- `__abstractmethods__` (*fastr.plugins.managers.networkmanager.NetworkManager attribute*), 294
- `__abstractmethods__` (*fastr.plugins.managers.objectmanager.ObjectManager attribute*), 295
- `__abstractmethods__` (*fastr.plugins.managers.pluginmanager.PluginManager attribute*), 297
- `__abstractmethods__` (*fastr.plugins.managers.pluginmanager.PluginSubManager attribute*), 298
- `__abstractmethods__` (*fastr.plugins.managers.pluginmanager.PluginsView attribute*), 299
- `__abstractmethods__` (*fastr.plugins.managers.targetmanager.TargetManager attribute*), 300
- `__abstractmethods__` (*fastr.plugins.managers.toolmanager.ToolManager attribute*), 300
- `__abstractmethods__` (*fastr.plugins.reportingplugin.ReportingPlugin attribute*), 290
- `__add__()` (*fastr.core.samples.SampleBaseId method*), 115
- `__add__()` (*fastr.core.samples.SampleItemBase method*), 120
- `__add__()` (*fastr.core.samples.SamplePayload method*), 123
- `__add__()` (*fastr.core.samples.SampleValue method*), 124
- `__add__()` (*fastr.helpers.configmanager.EmptyDefault method*), 204
- `__annotations__` (*fastr.core.resourcelimit.ResourceLimit attribute*), 112
- `__annotations__` (*fastr.core.samples.SampleBaseId at-*

tribute), 115  
 \_\_annotations\_\_ (fastr.core.samples.SampleValue attribute), 124  
 \_\_annotations\_\_ (fastr.datatypes.BaseDataType attribute), 140  
 \_\_annotations\_\_ (fastr.helpers.lockfile.DirectoryLock attribute), 211  
 \_\_args\_\_ (fastr.core.samples.SampleBaseId attribute), 115  
 \_\_args\_\_ (fastr.core.samples.SampleId attribute), 119  
 \_\_args\_\_ (fastr.core.samples.SampleIndex attribute), 119  
 \_\_args\_\_ (fastr.datatypes.DataTypeManager attribute), 143  
 \_\_args\_\_ (fastr.plugins.managers.executionpluginmanager.ExecutionPluginManager attribute), 290  
 \_\_args\_\_ (fastr.plugins.managers.interfacemanager.InterfacePluginManager attribute), 291  
 \_\_args\_\_ (fastr.plugins.managers.iopluginmanager.IOPluginManager attribute), 292  
 \_\_args\_\_ (fastr.plugins.managers.networkmanager.NetworkManager attribute), 294  
 \_\_args\_\_ (fastr.plugins.managers.objectmanager.ObjectManager attribute), 295  
 \_\_args\_\_ (fastr.plugins.managers.pluginmanager.PluginManager attribute), 297  
 \_\_args\_\_ (fastr.plugins.managers.pluginmanager.PluginSubManager attribute), 298  
 \_\_args\_\_ (fastr.plugins.managers.targetmanager.TargetManager attribute), 300  
 \_\_args\_\_ (fastr.plugins.managers.toolmanager.ToolManager attribute), 300  
 \_\_bool\_\_ () (fastr.execution.networkrun.NetworkRun method), 185  
 \_\_contains\_\_ () (fastr.core.samples.HasSamples method), 114  
 \_\_contains\_\_ () (fastr.core.samples.SampleCollection method), 116  
 \_\_contains\_\_ () (fastr.plugins.managers.objectmanager.ObjectManager method), 295  
 \_\_dataschemafile\_\_ (fastr.core.tool.Tool attribute), 128  
 \_\_dataschemafile\_\_ (fastr.execution.linkrun.LinkRun attribute), 179  
 \_\_dataschemafile\_\_ (fastr.execution.noderun.NodeRun attribute), 187  
 \_\_dataschemafile\_\_ (fastr.execution.sinknoderun.SinkNodeRun attribute), 190  
 \_\_dataschemafile\_\_ (fastr.execution.sourcenoderun.ConstantNodeRun attribute), 192  
 \_\_dataschemafile\_\_ (fastr.execution.sourcenoderun.SourceNodeRun attribute), 193  
 \_\_dataschemafile\_\_ (fastr.planning.link.Link attribute), 237  
 \_\_dataschemafile\_\_ (fastr.planning.network.Network attribute), 240  
 \_\_dataschemafile\_\_ (fastr.planning.node.ConstantNode attribute), 245  
 \_\_dataschemafile\_\_ (fastr.planning.node.Node attribute), 248  
 \_\_dataschemafile\_\_ (fastr.planning.node.SinkNode attribute), 251  
 \_\_dataschemafile\_\_ (fastr.planning.node.SourceNode attribute), 252  
 \_\_dataschemafile\_\_ (fastr.plugins.FastrInterface attribute), 261  
 \_\_dataschemafile\_\_ (fastr.plugins.FlowInterface attribute), 265  
 \_\_dataschemafile\_\_ (fastr.plugins.executionplugin.ExecutionPlugin attribute), 288  
 \_\_delitem\_\_ () (fastr.core.samples.SampleCollection method), 117  
 \_\_delitem\_\_ () (fastr.core.samples.SampleValue method), 124  
 \_\_delitem\_\_ () (fastr.helpers.configmanager.EmptyDefault method), 204  
 \_\_delitem\_\_ () (fastr.plugins.managers.pluginmanager.PluginsView method), 299  
 \_\_dict\_\_ (fastr.core.cardinality.CardinalitySpec attribute), 98  
 \_\_dict\_\_ (fastr.core.dimension.HasDimensions attribute), 105  
 \_\_dict\_\_ (fastr.core.interface.InputSpec attribute), 106  
 \_\_dict\_\_ (fastr.core.interface.InterfaceResult attribute), 108  
 \_\_dict\_\_ (fastr.core.interface.OutputSpec attribute), 108  
 \_\_dict\_\_ (fastr.core.provenance.Provenance attribute), 111  
 \_\_dict\_\_ (fastr.core.samples.SampleBaseId attribute), 117  
 \_\_dict\_\_ (fastr.core.samples.SampleCollection attribute), 117  
 \_\_dict\_\_ (fastr.core.samples.SampleItemBase attribute), 120  
 \_\_dict\_\_ (fastr.core.samples.SampleValue attribute), 124  
 \_\_dict\_\_ (fastr.core.target.ProcessUsageCollection attribute), 125  
 \_\_dict\_\_ (fastr.core.target.TargetResult attribute), 127  
 \_\_dict\_\_ (fastr.core.version.Version attribute), 131  
 \_\_dict\_\_ (fastr.core.vfs.VirtualFileSystem attribute), 134  
 \_\_dict\_\_ (fastr.execution.environmentmodules.EnvironmentModules attribute), 152  
 \_\_dict\_\_ (fastr.execution.networkanalyzer.NetworkAnalyzer

- `attribute`), 183
- `__dict__` (`fastr.execution.networkchunker.NetworkChunker` attribute), 184
- `__dict__` (`fastr.helpers.classproperty.ClassPropertyDescriptor` attribute), 198
- `__dict__` (`fastr.helpers.configmanager.Config` attribute), 201
- `__dict__` (`fastr.helpers.configmanager.EmptyDefault` attribute), 204
- `__dict__` (`fastr.helpers.filesynchelper.FileSyncHelper` attribute), 207
- `__dict__` (`fastr.helpers.lockfile.DirectoryLock` attribute), 211
- `__dict__` (`fastr.helpers.schematotable.SchemaPrinter` attribute), 212
- `__dict__` (`fastr.plugins.managers.pluginmanager.PluginsView` attribute), 299
- `__dict__` (`fastr.utils.cmd.upgrade.dummy_container` attribute), 308
- `__dir__` () (in module `fastr.__init__`), 85
- `__enter__` () (`fastr.core.target.Target` method), 127
- `__enter__` () (`fastr.helpers.lockfile.DirectoryLock` method), 211
- `__enter__` () (`fastr.plugins.DockerTarget` method), 258
- `__enter__` () (`fastr.plugins.LocalBinaryTarget` method), 269
- `__enter__` () (`fastr.plugins.SingularityTarget` method), 278
- `__enter__` () (`fastr.plugins.executionplugin.ExecutionPlugin` method), 288
- `__eq__` () (`fastr.api.ResourceLimit` method), 95
- `__eq__` () (`fastr.core.cardinality.AnyCardinalitySpec` method), 97
- `__eq__` () (`fastr.core.cardinality.AsCardinalitySpec` method), 98
- `__eq__` () (`fastr.core.cardinality.CardinalitySpec` method), 99
- `__eq__` () (`fastr.core.cardinality.ChoiceCardinalitySpec` method), 100
- `__eq__` () (`fastr.core.cardinality.IntCardinalitySpec` method), 100
- `__eq__` () (`fastr.core.cardinality.MaxCardinalitySpec` method), 101
- `__eq__` () (`fastr.core.cardinality.MinCardinalitySpec` method), 101
- `__eq__` () (`fastr.core.cardinality.RangeCardinalitySpec` method), 102
- `__eq__` () (`fastr.core.cardinality.ValueCardinalitySpec` method), 102
- `__eq__` () (`fastr.core.dimension.Dimension` method), 104
- `__eq__` () (`fastr.core.resourcelimit.ResourceLimit` method), 112
- `__eq__` () (`fastr.core.tool.Tool` method), 128
- `__eq__` () (`fastr.datatypes.BaseDataType` method), 140
- `__eq__` () (`fastr.datatypes.URLType` method), 149
- `__eq__` () (`fastr.execution.linkrun.LinkRun` method), 179
- `__eq__` () (`fastr.execution.networkrun.NetworkRun` method), 185
- `__eq__` () (`fastr.execution.noderun.NodeRun` method), 187
- `__eq__` () (`fastr.execution.sourcenoderun.SourceNodeRun` method), 193
- `__eq__` () (`fastr.planning.inputoutput.Input` method), 225
- `__eq__` () (`fastr.planning.inputoutput.Output` method), 229
- `__eq__` () (`fastr.planning.inputoutput.SubInput` method), 232
- `__eq__` () (`fastr.planning.inputoutput.SubOutput` method), 235
- `__eq__` () (`fastr.planning.link.Link` method), 237
- `__eq__` () (`fastr.planning.network.Network` method), 240
- `__eq__` () (`fastr.planning.node.MacroNode` method), 247
- `__eq__` () (`fastr.planning.node.Node` method), 248
- `__eq__` () (`fastr.plugins.FastrInterface` method), 261
- `__eq__` () (`fastr.plugins.FlowInterface` method), 265
- `__eq__` () (`fastr.plugins.NipypeInterface` method), 271
- `__exit__` () (`fastr.core.target.Target` method), 127
- `__exit__` () (`fastr.helpers.lockfile.DirectoryLock` method), 211
- `__exit__` () (`fastr.plugins.DockerTarget` method), 258
- `__exit__` () (`fastr.plugins.LocalBinaryTarget` method), 269
- `__exit__` () (`fastr.plugins.SingularityTarget` method), 278
- `__exit__` () (`fastr.plugins.executionplugin.ExecutionPlugin` method), 288
- `__extra__` (`fastr.core.samples.SampleBaseId` attribute), 115
- `__extra__` (`fastr.core.samples.SampleId` attribute), 119
- `__extra__` (`fastr.core.samples.SampleIndex` attribute), 119
- `__extra__` (`fastr.datatypes.DataTypeManager` attribute), 143
- `__extra__` (`fastr.plugins.managers.executionpluginmanager.ExecutionPluginManager` attribute), 291
- `__extra__` (`fastr.plugins.managers.interfacemanager.InterfacePluginManager` attribute), 291
- `__extra__` (`fastr.plugins.managers.iopluginmanager.IOPluginManager` attribute), 292
- `__extra__` (`fastr.plugins.managers.networkmanager.NetworkManager` attribute), 294
- `__extra__` (`fastr.plugins.managers.objectmanager.ObjectManager` attribute), 295
- `__extra__` (`fastr.plugins.managers.pluginmanager.PluginManager` attribute), 297
- `__extra__` (`fastr.plugins.managers.pluginmanager.PluginSubManager` attribute), 298
- `__extra__` (`fastr.plugins.managers.targetmanager.TargetManager` attribute), 299

attribute), 300  
 \_\_extra\_\_ (fastr.plugins.managers.toolmanager.ToolManager attribute), 300  
 \_\_format\_\_ (in module fastr.\_\_init\_\_), 85  
 \_\_get\_\_ (fastr.helpers.classproperty.ClassPropertyDescriptor method), 198  
 \_\_getattr\_\_ (fastr.helpers.lazy\_module.LazyModule method), 210  
 \_\_getitem\_\_ (fastr.api.Output method), 84  
 \_\_getitem\_\_ (fastr.core.samples.ContainsSamples method), 114  
 \_\_getitem\_\_ (fastr.core.samples.HasSamples method), 114  
 \_\_getitem\_\_ (fastr.core.samples.SampleCollection method), 117  
 \_\_getitem\_\_ (fastr.core.samples.SampleValue method), 124  
 \_\_getitem\_\_ (fastr.core.target.ProcessUsageCollection method), 125  
 \_\_getitem\_\_ (fastr.execution.inputoutputrun.InputRun method), 156  
 \_\_getitem\_\_ (fastr.execution.inputoutputrun.NamedSubinputRun method), 159  
 \_\_getitem\_\_ (fastr.execution.inputoutputrun.OutputRun method), 160  
 \_\_getitem\_\_ (fastr.execution.inputoutputrun.SourceOutputRun method), 162  
 \_\_getitem\_\_ (fastr.execution.inputoutputrun.SubInputRun method), 164  
 \_\_getitem\_\_ (fastr.execution.inputoutputrun.SubOutputRun method), 166  
 \_\_getitem\_\_ (fastr.execution.linkrun.LinkRun method), 179  
 \_\_getitem\_\_ (fastr.execution.networkrun.NetworkRun method), 185  
 \_\_getitem\_\_ (fastr.helpers.configmanager.EmptyDefault method), 205  
 \_\_getitem\_\_ (fastr.planning.inputgroup.InputGroup method), 216  
 \_\_getitem\_\_ (fastr.planning.inputoutput.Input method), 225  
 \_\_getitem\_\_ (fastr.planning.inputoutput.NamedSubInput method), 228  
 \_\_getitem\_\_ (fastr.planning.inputoutput.Output method), 229  
 \_\_getitem\_\_ (fastr.planning.inputoutput.SubInput method), 232  
 \_\_getitem\_\_ (fastr.planning.network.Network method), 240  
 \_\_getitem\_\_ (fastr.plugins.managers.objectmanager.ObjectManager method), 295  
 \_\_getitem\_\_ (fastr.plugins.managers.pluginmanager.PluginManager method), 299  
 \_\_getitem\_\_ (fastr.utils.cmd.upgrade.dummy\_container method), 308  
 \_\_getnewargs\_\_ (fastr.core.samples.SampleBaseId method), 115  
 \_\_getnewargs\_\_ (fastr.core.samples.SampleItemBase method), 121  
 \_\_getnewargs\_\_ (fastr.core.target.SystemUsageInfo method), 126  
 \_\_getnewargs\_\_ (fastr.utils.cmd.upgrade.FastrNamespaceType method), 308  
 \_\_getstate\_\_ (fastr.api.ResourceLimit method), 95  
 \_\_getstate\_\_ (fastr.core.interface.Interface method), 107  
 \_\_getstate\_\_ (fastr.core.resourcelimit.ResourceLimit method), 112  
 \_\_getstate\_\_ (fastr.core.samples.SampleValue method), 124  
 \_\_getstate\_\_ (fastr.core.tool.Tool method), 129  
 \_\_getstate\_\_ (fastr.datatypes.BaseDataType method), 140  
 \_\_getstate\_\_ (fastr.datatypes.Deferred method), 147  
 \_\_getstate\_\_ (fastr.execution.inputoutputrun.InputRun method), 157  
 \_\_getstate\_\_ (fastr.execution.inputoutputrun.OutputRun method), 160  
 \_\_getstate\_\_ (fastr.execution.inputoutputrun.SubInputRun method), 164  
 \_\_getstate\_\_ (fastr.execution.inputoutputrun.SubOutputRun method), 167  
 \_\_getstate\_\_ (fastr.execution.job.Job method), 169  
 \_\_getstate\_\_ (fastr.execution.job.SinkJob method), 176  
 \_\_getstate\_\_ (fastr.execution.job.SourceJob method), 178  
 \_\_getstate\_\_ (fastr.execution.linkrun.LinkRun method), 180  
 \_\_getstate\_\_ (fastr.execution.macronoderun.MacroNodeRun method), 182  
 \_\_getstate\_\_ (fastr.execution.networkrun.NetworkRun method), 185  
 \_\_getstate\_\_ (fastr.execution.noderun.NodeRun method), 188  
 \_\_getstate\_\_ (fastr.execution.sinknoderun.SinkNodeRun method), 190  
 \_\_getstate\_\_ (fastr.execution.sourcenoderun.ConstantNodeRun method), 192  
 \_\_getstate\_\_ (fastr.execution.sourcenoderun.SourceNodeRun method), 193  
 \_\_getstate\_\_ (fastr.planning.inputoutput.BaseInputOutput method), 222  
 \_\_getstate\_\_ (fastr.planning.inputoutput.Input method), 225  
 \_\_getstate\_\_ (fastr.planning.inputoutput.Output method), 230  
 \_\_getstate\_\_ (fastr.planning.inputoutput.SubInput



- method*), 232
- `__getstate__()` (*fastr.planning.inputoutput.SubOutput method*), 235
- `__getstate__()` (*fastr.planning.link.Link method*), 237
- `__getstate__()` (*fastr.planning.network.Network method*), 240
- `__getstate__()` (*fastr.planning.node.ConstantNode method*), 245
- `__getstate__()` (*fastr.planning.node.MacroNode method*), 247
- `__getstate__()` (*fastr.planning.node.Node method*), 248
- `__getstate__()` (*fastr.planning.node.SinkNode method*), 251
- `__getstate__()` (*fastr.planning.node.SourceNode method*), 252
- `__getstate__()` (*fastr.plugins.FastrInterface method*), 261
- `__getstate__()` (*fastr.plugins.FlowInterface method*), 265
- `__getstate__()` (*fastr.plugins.NipypeInterface method*), 271
- `__hash__` (*fastr.api.ResourceLimit attribute*), 95
- `__hash__` (*fastr.core.cardinality.AnyCardinalitySpec attribute*), 97
- `__hash__` (*fastr.core.cardinality.AsCardinalitySpec attribute*), 98
- `__hash__` (*fastr.core.cardinality.CardinalitySpec attribute*), 99
- `__hash__` (*fastr.core.cardinality.ChoiceCardinalitySpec attribute*), 100
- `__hash__` (*fastr.core.cardinality.IntCardinalitySpec attribute*), 100
- `__hash__` (*fastr.core.cardinality.MaxCardinalitySpec attribute*), 101
- `__hash__` (*fastr.core.cardinality.MinCardinalitySpec attribute*), 101
- `__hash__` (*fastr.core.cardinality.RangeCardinalitySpec attribute*), 102
- `__hash__` (*fastr.core.cardinality.ValueCardinalitySpec attribute*), 102
- `__hash__` (*fastr.core.dimension.Dimension attribute*), 104
- `__hash__` (*fastr.core.resourcelimit.ResourceLimit attribute*), 112
- `__hash__` (*fastr.core.tool.Tool attribute*), 129
- `__hash__` (*fastr.datatypes.BaseDataType attribute*), 140
- `__hash__` (*fastr.datatypes.URLType attribute*), 150
- `__hash__` (*fastr.execution.linkrun.LinkRun attribute*), 180
- `__hash__` (*fastr.execution.networkrun.NetworkRun attribute*), 185
- `__hash__` (*fastr.execution.noderun.NodeRun attribute*), 188
- `__hash__` (*fastr.execution.sourcenoderun.SourceNodeRun attribute*), 194
- `__hash__` (*fastr.planning.inputoutput.Input attribute*), 225
- `__hash__` (*fastr.planning.inputoutput.Output attribute*), 230
- `__hash__` (*fastr.planning.inputoutput.SubInput attribute*), 233
- `__hash__` (*fastr.planning.inputoutput.SubOutput attribute*), 235
- `__hash__` (*fastr.planning.link.Link attribute*), 237
- `__hash__` (*fastr.planning.network.Network attribute*), 240
- `__hash__` (*fastr.planning.node.MacroNode attribute*), 247
- `__hash__` (*fastr.planning.node.Node attribute*), 248
- `__hash__` (*fastr.plugins.FastrInterface attribute*), 261
- `__hash__` (*fastr.plugins.FlowInterface attribute*), 265
- `__hash__` (*fastr.plugins.NipypeInterface attribute*), 271
- `__iadd__()` (*fastr.helpers.configmanager.EmptyDefault method*), 205
- `__init__()` (*fastr.api.ResourceLimit method*), 95
- `__init__()` (*fastr.core.cardinality.AsCardinalitySpec method*), 98
- `__init__()` (*fastr.core.cardinality.CardinalitySpec method*), 99
- `__init__()` (*fastr.core.cardinality.ChoiceCardinalitySpec method*), 100
- `__init__()` (*fastr.core.cardinality.IntCardinalitySpec method*), 100
- `__init__()` (*fastr.core.cardinality.MaxCardinalitySpec method*), 101
- `__init__()` (*fastr.core.cardinality.MinCardinalitySpec method*), 101
- `__init__()` (*fastr.core.cardinality.RangeCardinalitySpec method*), 102
- `__init__()` (*fastr.core.cardinality.ValueCardinalitySpec method*), 102
- `__init__()` (*fastr.core.dimension.Dimension method*), 104
- `__init__()` (*fastr.core.interface.InterfaceResult method*), 108
- `__init__()` (*fastr.core.ioplugin.IOPlugin method*), 109
- `__init__()` (*fastr.core.provenance.Provenance method*), 111
- `__init__()` (*fastr.core.resourcelimit.ResourceLimit method*), 112
- `__init__()` (*fastr.core.samples.SampleCollection method*), 117
- `__init__()` (*fastr.core.samples.SampleValue method*), 124
- `__init__()` (*fastr.core.target.ProcessUsageCollection method*), 126
- `__init__()` (*fastr.core.target.TargetResult method*), 128

`__init__()` (*fastr.core.tool.Tool* method), 129

`__init__()` (*fastr.core.vfs.VirtualFileSystem* method), 135

`__init__()` (*fastr.datatypes.BaseDataType* method), 140

`__init__()` (*fastr.datatypes.DataType* method), 143

`__init__()` (*fastr.datatypes.DataTypeManager* method), 144

`__init__()` (*fastr.datatypes.Deferred* method), 147

`__init__()` (*fastr.datatypes.EnumType* method), 148

`__init__()` (*fastr.datatypes.Missing* method), 148

`__init__()` (*fastr.datatypes.TypeGroup* method), 149

`__init__()` (*fastr.datatypes.URLType* method), 150

`__init__()` (*fastr.datatypes.ValueType* method), 151

`__init__()` (*fastr.exceptions.FastrError* method), 87

`__init__()` (*fastr.exceptions.FastrExecutableNotFoundError* method), 87

`__init__()` (*fastr.exceptions.FastrFileNotFoundError* method), 88

`__init__()` (*fastr.exceptions.FastrLockNotAcquired* method), 89

`__init__()` (*fastr.exceptions.FastrScriptNotFoundError* method), 91

`__init__()` (*fastr.exceptions.FastrSerializationError* method), 91

`__init__()` (*fastr.execution.environmentmodules.EnvironmentModule* method), 152

`__init__()` (*fastr.execution.inputoutputrun.BaseInputRun* method), 156

`__init__()` (*fastr.execution.inputoutputrun.InputRun* method), 157

`__init__()` (*fastr.execution.inputoutputrun.NamedSubInputRun* method), 159

`__init__()` (*fastr.execution.inputoutputrun.OutputRun* method), 161

`__init__()` (*fastr.execution.inputoutputrun.SourceOutputRun* method), 163

`__init__()` (*fastr.execution.inputoutputrun.SubInputRun* method), 164

`__init__()` (*fastr.execution.inputoutputrun.SubOutputRun* method), 167

`__init__()` (*fastr.execution.job.InlineJob* method), 168

`__init__()` (*fastr.execution.job.Job* method), 169

`__init__()` (*fastr.execution.job.JobState* method), 175

`__init__()` (*fastr.execution.job.SinkJob* method), 176

`__init__()` (*fastr.execution.job.SourceJob* method), 178

`__init__()` (*fastr.execution.linkrun.LinkRun* method), 180

`__init__()` (*fastr.execution.macronoderun.MacroNodeRun* method), 182

`__init__()` (*fastr.execution.networkchunker.DefaultNetworkChunker* method), 184

`__init__()` (*fastr.execution.networkrun.NetworkRun* method), 185

`__init__()` (*fastr.execution.noderun.NodeRun* method), 188

`__init__()` (*fastr.execution.sinknoderun.SinkNodeRun* method), 190

`__init__()` (*fastr.execution.sourcenoderun.ConstantNodeRun* method), 192

`__init__()` (*fastr.execution.sourcenoderun.SourceNodeRun* method), 194

`__init__()` (*fastr.helpers.classproperty.ClassPropertyDescriptor* method), 198

`__init__()` (*fastr.helpers.configmanager.Config* method), 203

`__init__()` (*fastr.helpers.configmanager.EmptyDefault* method), 205

`__init__()` (*fastr.helpers.filesynchelper.FileSyncHelper* method), 207

`__init__()` (*fastr.helpers.jsonschemaparser.FastrRefResolver* method), 208

`__init__()` (*fastr.helpers.lazy\_module.LazyModule* method), 210

`__init__()` (*fastr.helpers.lockfile.DirectoryLock* method), 211

`__init__()` (*fastr.helpers.schematotable.SchemaPrinter* method), 213

`__init__()` (*fastr.planning.inputgroup.InputGroup* method), 216

`__init__()` (*fastr.planning.inputgroupcombiner.BaseInputGroupCombine* method), 218

`__init__()` (*fastr.planning.inputgroupcombiner.MergingInputGroupComb* method), 220

`__init__()` (*fastr.planning.inputoutput.BaseInput* method), 221

`__init__()` (*fastr.planning.inputoutput.BaseInputOutput* method), 222

`__init__()` (*fastr.planning.inputoutput.BaseOutput* method), 224

`__init__()` (*fastr.planning.inputoutput.Input* method), 225

`__init__()` (*fastr.planning.inputoutput.NamedSubInput* method), 229

`__init__()` (*fastr.planning.inputoutput.Output* method), 230

`__init__()` (*fastr.planning.inputoutput.SourceOutput* method), 231

`__init__()` (*fastr.planning.inputoutput.SubInput* method), 233

`__init__()` (*fastr.planning.inputoutput.SubOutput* method), 235

`__init__()` (*fastr.planning.link.Link* method), 237

`__init__()` (*fastr.planning.network.Network* method), 240

`__init__()` (*fastr.planning.node.ConstantNode* method), 245

`__init__()` (*fastr.planning.node.FlowNode* method),

[246](#)  
`__init__()` (*fastr.planning.node.MacroNode* method), [247](#)  
`__init__()` (*fastr.planning.node.Node* method), [248](#)  
`__init__()` (*fastr.planning.node.SinkNode* method), [251](#)  
`__init__()` (*fastr.planning.node.SourceNode* method), [252](#)  
`__init__()` (*fastr.plugins.BlockingExecution* method), [253](#)  
`__init__()` (*fastr.plugins.CommaSeperatedValueFile* method), [255](#)  
`__init__()` (*fastr.plugins.DRMAAExecution* method), [257](#)  
`__init__()` (*fastr.plugins.DockerTarget* method), [258](#)  
`__init__()` (*fastr.plugins.ElasticsearchReporter* method), [259](#)  
`__init__()` (*fastr.plugins.FastrInterface* method), [261](#)  
`__init__()` (*fastr.plugins.FileSystem* method), [263](#)  
`__init__()` (*fastr.plugins.FlowInterface* method), [265](#)  
`__init__()` (*fastr.plugins.HTTPPlugin* method), [266](#)  
`__init__()` (*fastr.plugins.LinearExecution* method), [267](#)  
`__init__()` (*fastr.plugins.LocalBinaryTarget* method), [269](#)  
`__init__()` (*fastr.plugins.MacroTarget* method), [270](#)  
`__init__()` (*fastr.plugins.NetworkScope* method), [270](#)  
`__init__()` (*fastr.plugins.NipypeInterface* method), [271](#)  
`__init__()` (*fastr.plugins.Null* method), [272](#)  
`__init__()` (*fastr.plugins.PimReporter* method), [273](#)  
`__init__()` (*fastr.plugins.ProcessPoolExecution* method), [274](#)  
`__init__()` (*fastr.plugins.RQExecution* method), [275](#)  
`__init__()` (*fastr.plugins.Reference* method), [276](#)  
`__init__()` (*fastr.plugins.S3Filesystem* method), [276](#)  
`__init__()` (*fastr.plugins.SingularityTarget* method), [278](#)  
`__init__()` (*fastr.plugins.SlurmExecution* method), [279](#)  
`__init__()` (*fastr.plugins.StrongrExecution* method), [280](#)  
`__init__()` (*fastr.plugins.VirtualFileSystemRegularExpression* method), [282](#)  
`__init__()` (*fastr.plugins.VirtualFileSystemValueList* method), [283](#)  
`__init__()` (*fastr.plugins.XNATStorage* method), [285](#)  
`__init__()` (*fastr.plugins.executionplugin.ExecutionPlugin* method), [288](#)  
`__init__()` (*fastr.plugins.managers.executionpluginmanager.ExecutionPluginManager* method), [291](#)  
`__init__()` (*fastr.plugins.managers.interfacemanager.InterfacePluginManager* method), [291](#)  
`__init__()` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), [292](#)  
`__init__()` (*fastr.plugins.managers.objectmanager.ObjectManager* method), [295](#)  
`__init__()` (*fastr.plugins.managers.pluginmanager.PluginManager* method), [297](#)  
`__init__()` (*fastr.plugins.managers.pluginmanager.PluginSubManager* method), [298](#)  
`__init__()` (*fastr.plugins.managers.pluginmanager.PluginsView* method), [299](#)  
`__init__()` (*fastr.plugins.managers.targetmanager.TargetManager* method), [300](#)  
`__init_subclass__()` (*fastr.execution.basenoderun.BaseNodeRun* class method), [152](#)  
`__init_subclass__()` (*fastr.planning.node.BaseNode* class method), [245](#)  
`__init_subclass__()` (in module *fastr.\_\_init\_\_*), [85](#)  
`__iter__()` (*fastr.core.samples.HasSamples* method), [114](#)  
`__iter__()` (*fastr.core.samples.SampleCollection* method), [117](#)  
`__iter__()` (*fastr.core.samples.SampleValue* method), [124](#)  
`__iter__()` (*fastr.planning.inputgroupcombiner.BaseInputGroupCombine* method), [218](#)  
`__iter__()` (*fastr.planning.inputoutput.BaseInputOutput* method), [222](#)  
`__iter__()` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), [292](#)  
`__iter__()` (*fastr.plugins.managers.pluginmanager.PluginsView* method), [299](#)  
`__keytransform__()` (*fastr.datatypes.DataTypeManager* method), [144](#)  
`__keytransform__()` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), [292](#)  
`__keytransform__()` (*fastr.plugins.managers.objectmanager.ObjectManager* method), [295](#)  
`__len__()` (*fastr.core.samples.SampleCollection* method), [118](#)  
`__len__()` (*fastr.core.samples.SampleValue* method), [124](#)  
`__len__()` (*fastr.core.target.ProcessUsageCollection* method), [126](#)  
`__len__()` (*fastr.execution.inputoutputrun.SubOutputRun* method), [167](#)  
`__len__()` (*fastr.planning.inputoutput.SubOutput* method), [235](#)  
`__len__()` (*fastr.plugins.managers.pluginmanager.PluginsView* method), [299](#)  
`__len__()` (*fastr.plugins.managers.pluginmanager.PluginsView* method), [299](#)  
`__lshift__()` (*fastr.planning.inputoutput.BaseInput* method), [221](#)  
`__module__` (*fastr.api.ResourceLimit* attribute), [96](#)  
`__module__` (*fastr.core.cardinality.AnyCardinalitySpec* attribute), [97](#)  
`__module__` (*fastr.core.cardinality.AsCardinalitySpec* attribute), [98](#)

`__module__` (*fastr.core.cardinality.CardinalitySpec* attribute), 99

`__module__` (*fastr.core.cardinality.ChoiceCardinalitySpec* attribute), 100

`__module__` (*fastr.core.cardinality.IntCardinalitySpec* attribute), 100

`__module__` (*fastr.core.cardinality.MaxCardinalitySpec* attribute), 101

`__module__` (*fastr.core.cardinality.MinCardinalitySpec* attribute), 102

`__module__` (*fastr.core.cardinality.RangeCardinalitySpec* attribute), 102

`__module__` (*fastr.core.cardinality.ValueCardinalitySpec* attribute), 102

`__module__` (*fastr.core.dimension.Dimension* attribute), 104

`__module__` (*fastr.core.dimension.ForwardsDimensions* attribute), 105

`__module__` (*fastr.core.dimension.HasDimensions* attribute), 105

`__module__` (*fastr.core.interface.InputSpec* attribute), 106

`__module__` (*fastr.core.interface.Interface* attribute), 107

`__module__` (*fastr.core.interface.InterfaceResult* attribute), 108

`__module__` (*fastr.core.interface.OutputSpec* attribute), 108

`__module__` (*fastr.core.ioplugin.IOPlugin* attribute), 109

`__module__` (*fastr.core.provenance.Provenance* attribute), 111

`__module__` (*fastr.core.resourcelimit.ResourceLimit* attribute), 113

`__module__` (*fastr.core.samples.ContainsSamples* attribute), 114

`__module__` (*fastr.core.samples.HasSamples* attribute), 114

`__module__` (*fastr.core.samples.SampleBaseId* attribute), 115

`__module__` (*fastr.core.samples.SampleCollection* attribute), 118

`__module__` (*fastr.core.samples.SampleId* attribute), 119

`__module__` (*fastr.core.samples.SampleIndex* attribute), 119

`__module__` (*fastr.core.samples.SampleItem* attribute), 120

`__module__` (*fastr.core.samples.SampleItemBase* attribute), 121

`__module__` (*fastr.core.samples.SamplePayload* attribute), 123

`__module__` (*fastr.core.samples.SampleState* attribute), 124

`__module__` (*fastr.core.samples.SampleValue* attribute), 124

`__module__` (*fastr.core.target.ProcessUsageCollection* attribute), 126

`__module__` (*fastr.core.target.SubprocessBasedTarget* attribute), 126

`__module__` (*fastr.core.target.SystemUsageInfo* attribute), 126

`__module__` (*fastr.core.target.Target* attribute), 127

`__module__` (*fastr.core.target.TargetResult* attribute), 128

`__module__` (*fastr.core.tool.Tool* attribute), 129

`__module__` (*fastr.core.version.Version* attribute), 131

`__module__` (*fastr.core.vfs.VirtualFileSystem* attribute), 135

`__module__` (*fastr.datatypes.AnyFile* attribute), 139

`__module__` (*fastr.datatypes.AnyType* attribute), 140

`__module__` (*fastr.datatypes.BaseDataType* attribute), 141

`__module__` (*fastr.datatypes.DataType* attribute), 143

`__module__` (*fastr.datatypes.DataTypeManager* attribute), 144

`__module__` (*fastr.datatypes.Deferred* attribute), 147

`__module__` (*fastr.datatypes.EnumType* attribute), 148

`__module__` (*fastr.datatypes.Missing* attribute), 149

`__module__` (*fastr.datatypes.TypeGroup* attribute), 149

`__module__` (*fastr.datatypes.URLType* attribute), 150

`__module__` (*fastr.datatypes.ValueType* attribute), 151

`__module__` (*fastr.exceptions.FastrAttributeError* attribute), 86

`__module__` (*fastr.exceptions.FastrCannotChangeAttributeError* attribute), 86

`__module__` (*fastr.exceptions.FastrCardinalityError* attribute), 86

`__module__` (*fastr.exceptions.FastrCollectorError* attribute), 86

`__module__` (*fastr.exceptions.FastrDataTypeFileNotReadable* attribute), 86

`__module__` (*fastr.exceptions.FastrDataTypeMismatchError* attribute), 86

`__module__` (*fastr.exceptions.FastrDataTypeNotAvailableError* attribute), 86

`__module__` (*fastr.exceptions.FastrDataTypeNotInstantiableError* attribute), 86

`__module__` (*fastr.exceptions.FastrDataTypeValueError* attribute), 87

`__module__` (*fastr.exceptions.FastrError* attribute), 87

`__module__` (*fastr.exceptions.FastrErrorInSubprocess* attribute), 87

`__module__` (*fastr.exceptions.FastrExecutableNotFoundError* attribute), 87

`__module__` (*fastr.exceptions.FastrExecutionError* attribute), 88

`__module__` (*fastr.exceptions.FastrFileNotFound* attribute), 88

`__module__` (*fastr.exceptions.FastrIOError* attribute), 88

`__module__` (*fastr.exceptions.FastrImportError* at-



tribute), 88

\_\_module\_\_ (fastr.exceptions.FastrIndexError attribute), 88

\_\_module\_\_ (fastr.exceptions.FastrIndexNonexistent attribute), 88

\_\_module\_\_ (fastr.exceptions.FastrKeyError attribute), 88

\_\_module\_\_ (fastr.exceptions.FastrLockNotAcquired attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrLookupError attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrMountUnknownError attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrNetworkMismatchError attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrNetworkUnknownError attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrNoValidTargetError attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrNodeAlreadyPreparedError attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrNodeNotPreparedError attribute), 89

\_\_module\_\_ (fastr.exceptions.FastrNodeNotValidError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrNotExecutableError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrNotImplementedError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrOSError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrObjectUnknownError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrOptionalModuleNotAvailableError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrOutputValidationError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrParentMismatchError attribute), 90

\_\_module\_\_ (fastr.exceptions.FastrPluginCapabilityNotImplementedError attribute), 91

\_\_module\_\_ (fastr.exceptions.FastrPluginNotAvailable attribute), 91

\_\_module\_\_ (fastr.exceptions.FastrPluginNotLoaded attribute), 91

\_\_module\_\_ (fastr.exceptions.FastrResultFileNotFound attribute), 91

\_\_module\_\_ (fastr.exceptions.FastrScriptNotFoundError attribute), 91

\_\_module\_\_ (fastr.exceptions.FastrSerializationError attribute), 91

\_\_module\_\_ (fastr.exceptions.FastrSerializationIgnoreDefault attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrSerializationInvalidDataError attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrSerializationMethodError attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrSinkDataUnavailableError attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrSizeInvalidError attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrSizeMismatchError attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrSizeUnknownError attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrSourceDataUnavailableError attribute), 92

\_\_module\_\_ (fastr.exceptions.FastrStateError attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrSubprocessNotFinished attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrToolNotAvailableError attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrToolTargetNotFound attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrToolUnknownError attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrToolVersionError attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrTypeError attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrUnknownURLSchemeError attribute), 93

\_\_module\_\_ (fastr.exceptions.FastrValueError attribute), 94

\_\_module\_\_ (fastr.exceptions.FastrVersionInvalidError attribute), 94

\_\_module\_\_ (fastr.exceptions.FastrVersionMismatchError attribute), 94

\_\_module\_\_ (fastr.execution.basenoderun.BaseNodeRun attribute), 152

\_\_module\_\_ (fastr.execution.environmentmodules.EnvironmentModules attribute), 152

\_\_module\_\_ (fastr.execution.environmentmodules.ModuleSystem attribute), 154

\_\_module\_\_ (fastr.execution.flownoderun.AdvancedFlowNodeRun attribute), 154

\_\_module\_\_ (fastr.execution.flownoderun.FlowNodeRun attribute), 155

\_\_module\_\_ (fastr.execution.inputoutputputrun.AdvancedFlowOutputRun attribute), 155

\_\_module\_\_ (fastr.execution.inputoutputputrun.BaseInputRun attribute), 156

\_\_module\_\_ (fastr.execution.inputoutputputrun.InputRun attribute), 157

\_\_module\_\_ (fastr.execution.inputoutputputrun.MacroOutputRun attribute), 159

\_\_module\_\_ (fastr.execution.inputoutputputrun.NamedSubinputRun attribute), 159

- `attribute`), 160
- `__module__` (`fastr.execution.inputoutputrun.OutputRun` `attribute`), 161
- `__module__` (`fastr.execution.inputoutputrun.SourceOutputRun` `attribute`), 163
- `__module__` (`fastr.execution.inputoutputrun.SubInputRun` `attribute`), 165
- `__module__` (`fastr.execution.inputoutputrun.SubOutputRun` `attribute`), 167
- `__module__` (`fastr.execution.job.InlineJob` `attribute`), 169
- `__module__` (`fastr.execution.job.Job` `attribute`), 170
- `__module__` (`fastr.execution.job.JobCleanupLevel` `attribute`), 174
- `__module__` (`fastr.execution.job.JobState` `attribute`), 175
- `__module__` (`fastr.execution.job.SinkJob` `attribute`), 177
- `__module__` (`fastr.execution.job.SourceJob` `attribute`), 178
- `__module__` (`fastr.execution.linkrun.LinkRun` `attribute`), 180
- `__module__` (`fastr.execution.macronoderun.MacroNodeRun` `attribute`), 182
- `__module__` (`fastr.execution.networkanalyzer.DefaultNetworkAnalyzer` `attribute`), 183
- `__module__` (`fastr.execution.networkanalyzer.NetworkAnalyzer` `attribute`), 183
- `__module__` (`fastr.execution.networkchunker.DefaultNetworkChunker` `attribute`), 184
- `__module__` (`fastr.execution.networkchunker.NetworkChunker` `attribute`), 184
- `__module__` (`fastr.execution.networkrun.NetworkRun` `attribute`), 186
- `__module__` (`fastr.execution.noderun.NodeRun` `attribute`), 188
- `__module__` (`fastr.execution.sinknoderun.SinkNodeRun` `attribute`), 191
- `__module__` (`fastr.execution.sourcenoderun.ConstantNodeRun` `attribute`), 193
- `__module__` (`fastr.execution.sourcenoderun.SourceNodeRun` `attribute`), 194
- `__module__` (`fastr.helpers.classproperty.ClassPropertyDescriptor` `attribute`), 198
- `__module__` (`fastr.helpers.configmanager.Config` `attribute`), 203
- `__module__` (`fastr.helpers.configmanager.EmptyDefault` `attribute`), 205
- `__module__` (`fastr.helpers.configmanager.FastrLogRecordFilter` `attribute`), 205
- `__module__` (`fastr.helpers.events.EventType` `attribute`), 206
- `__module__` (`fastr.helpers.events.FastrLogEventHandler` `attribute`), 206
- `__module__` (`fastr.helpers.filesynchelper.FileSyncHelper` `attribute`), 207
- `__module__` (`fastr.helpers.jsonschemaparser.FastrRefResolver` `attribute`), 208
- `__module__` (`fastr.helpers.lazy_module.LazyModule` `attribute`), 210
- `__module__` (`fastr.helpers.lockfile.DirectoryLock` `attribute`), 211
- `__module__` (`fastr.helpers.schematatable.SchemaPrinter` `attribute`), 213
- `__module__` (`fastr.planning.inputgroup.InputGroup` `attribute`), 216
- `__module__` (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` `attribute`), 218
- `__module__` (`fastr.planning.inputgroupcombiner.DefaultInputGroupCombiner` `attribute`), 219
- `__module__` (`fastr.planning.inputgroupcombiner.MergingInputGroupCombiner` `attribute`), 220
- `__module__` (`fastr.planning.inputoutput.AdvancedFlowOutput` `attribute`), 221
- `__module__` (`fastr.planning.inputoutput.BaseInput` `attribute`), 221
- `__module__` (`fastr.planning.inputoutput.BaseInputOutput` `attribute`), 223
- `__module__` (`fastr.planning.inputoutput.BaseOutput` `attribute`), 224
- `__module__` (`fastr.planning.inputoutput.Input` `attribute`), 225
- `__module__` (`fastr.planning.inputoutput.MacroInput` `attribute`), 228
- `__module__` (`fastr.planning.inputoutput.MacroOutput` `attribute`), 228
- `__module__` (`fastr.planning.inputoutput.NamedSubInput` `attribute`), 229
- `__module__` (`fastr.planning.inputoutput.Output` `attribute`), 230
- `__module__` (`fastr.planning.inputoutput.SourceOutput` `attribute`), 232
- `__module__` (`fastr.planning.inputoutput.SubInput` `attribute`), 233
- `__module__` (`fastr.planning.inputoutput.SubOutput` `attribute`), 235
- `__module__` (`fastr.planning.link.Link` `attribute`), 238
- `__module__` (`fastr.planning.network.Network` `attribute`), 241
- `__module__` (`fastr.planning.node.AdvancedFlowNode` `attribute`), 244
- `__module__` (`fastr.planning.node.BaseNode` `attribute`), 245
- `__module__` (`fastr.planning.node.ConstantNode` `attribute`), 246
- `__module__` (`fastr.planning.node.FlowNode` `attribute`), 246
- `__module__` (`fastr.planning.node.InputDict` `attribute`), 247
- `__module__` (`fastr.planning.node.MacroNode` `attribute`), 247

- `__module__` (*fastr.planning.node.Node* attribute), 249
- `__module__` (*fastr.planning.node.OutputDict* attribute), 251
- `__module__` (*fastr.planning.node.SinkNode* attribute), 251
- `__module__` (*fastr.planning.node.SourceNode* attribute), 252
- `__module__` (*fastr.plugins.BlockingExecution* attribute), 254
- `__module__` (*fastr.plugins.CommaSeperatedValueFile* attribute), 255
- `__module__` (*fastr.plugins.CrossValidation* attribute), 255
- `__module__` (*fastr.plugins.DRMAAExecution* attribute), 257
- `__module__` (*fastr.plugins.DockerTarget* attribute), 258
- `__module__` (*fastr.plugins.ElasticsearchReporter* attribute), 259
- `__module__` (*fastr.plugins.FastrInterface* attribute), 262
- `__module__` (*fastr.plugins.FileSystem* attribute), 264
- `__module__` (*fastr.plugins.FlowInterface* attribute), 265
- `__module__` (*fastr.plugins.HTTPPlugin* attribute), 266
- `__module__` (*fastr.plugins.LinearExecution* attribute), 267
- `__module__` (*fastr.plugins.LocalBinaryTarget* attribute), 269
- `__module__` (*fastr.plugins.MacroTarget* attribute), 270
- `__module__` (*fastr.plugins.NetworkScope* attribute), 270
- `__module__` (*fastr.plugins.NipypeInterface* attribute), 271
- `__module__` (*fastr.plugins.Null* attribute), 272
- `__module__` (*fastr.plugins.PimReporter* attribute), 273
- `__module__` (*fastr.plugins.ProcessPoolExecution* attribute), 274
- `__module__` (*fastr.plugins.RQExecution* attribute), 275
- `__module__` (*fastr.plugins.Reference* attribute), 276
- `__module__` (*fastr.plugins.S3Filesystem* attribute), 276
- `__module__` (*fastr.plugins.SimpleReport* attribute), 278
- `__module__` (*fastr.plugins.SingularityTarget* attribute), 278
- `__module__` (*fastr.plugins.SlurmExecution* attribute), 279
- `__module__` (*fastr.plugins.StrongrExecution* attribute), 280
- `__module__` (*fastr.plugins.VirtualFileSystem* attribute), 281
- `__module__` (*fastr.plugins.VirtualFileSystemRegularExpression* attribute), 282
- `__module__` (*fastr.plugins.VirtualFileSystemValueList* attribute), 283
- `__module__` (*fastr.plugins.XNATStorage* attribute), 285
- `__module__` (*fastr.plugins.executionplugin.ExecutionPlugin* attribute), 288
- `__module__` (*fastr.plugins.executionplugin.JobAction* attribute), 290
- `__module__` (*fastr.plugins.managers.executionpluginmanager.ExecutionPluginManager* attribute), 291
- `__module__` (*fastr.plugins.managers.interfacemanager.InterfacePluginManager* attribute), 291
- `__module__` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* attribute), 292
- `__module__` (*fastr.plugins.managers.networkmanager.NetworkManager* attribute), 294
- `__module__` (*fastr.plugins.managers.objectmanager.ObjectManager* attribute), 296
- `__module__` (*fastr.plugins.managers.pluginmanager.PluginManager* attribute), 297
- `__module__` (*fastr.plugins.managers.pluginmanager.PluginSubManager* attribute), 298
- `__module__` (*fastr.plugins.managers.pluginmanager.PluginsView* attribute), 299
- `__module__` (*fastr.plugins.managers.targetmanager.TargetManager* attribute), 300
- `__module__` (*fastr.plugins.managers.toolmanager.ToolManager* attribute), 300
- `__module__` (*fastr.plugins.reportingplugin.ReportingPlugin* attribute), 290
- `__module__` (*fastr.utils.cmd.upgrade.FastrNamespaceType* attribute), 308
- `__module__` (*fastr.utils.cmd.upgrade.dummy\_container* attribute), 308
- `__ne__()` (*fastr.api.ResourceLimit* method), 96
- `__ne__()` (*fastr.core.cardinality.CardinalitySpec* method), 99
- `__ne__()` (*fastr.core.dimension.Dimension* method), 104
- `__ne__()` (*fastr.core.resourcelimit.ResourceLimit* method), 113
- `__ne__()` (*fastr.datatypes.BaseDataType* method), 141
- `__ne__()` (*fastr.execution.networkrun.NetworkRun* method), 186
- `__ne__()` (*fastr.planning.inputoutput.BaseInputOutput* method), 223
- `__ne__()` (*fastr.planning.network.Network* method), 241
- `__ne__()` (*fastr.planning.node.Node* method), 249
- `__new__()` (*fastr.core.interface.InputSpec* static method), 106
- `__new__()` (*fastr.core.interface.OutputSpec* static method), 108
- `__new__()` (*fastr.core.samples.SampleBaseId* static method), 115
- `__new__()` (*fastr.core.samples.SampleItem* static method), 120
- `__new__()` (*fastr.core.samples.SampleItemBase* static method), 121
- `__new__()` (*fastr.core.samples.SamplePayload* static method), 123
- `__new__()` (*fastr.core.target.SystemUsageInfo* static method), 126





- `attribute`), 300
- `__parameters__` (*fastr.plugins.managers.toolmanager.ToolManager* method), 223
- `attribute`), 301
- `__radd__` () (*fastr.core.samples.SampleBaseId* method), 116
- `__radd__` () (*fastr.core.samples.SampleValue* method), 124
- `__radd__` () (*fastr.helpers.configmanager.EmptyDefault* method), 205
- `__reduce__` () (in module *fastr.\_\_init\_\_*), 85
- `__reduce_ex__` () (*fastr.datatypes.BaseDataType* method), 141
- `__reduce_ex__` () (*fastr.datatypes.EnumType* method), 148
- `__reduce_ex__` () (in module *fastr.\_\_init\_\_*), 85
- `__repr__` () (*fastr.core.cardinality.CardinalitySpec* method), 99
- `__repr__` () (*fastr.core.dimension.Dimension* method), 104
- `__repr__` () (*fastr.core.samples.SampleBaseId* method), 116
- `__repr__` () (*fastr.core.samples.SampleCollection* method), 118
- `__repr__` () (*fastr.core.samples.SampleIndex* method), 119
- `__repr__` () (*fastr.core.samples.SampleItemBase* method), 121
- `__repr__` () (*fastr.core.samples.SampleValue* method), 125
- `__repr__` () (*fastr.core.target.SystemUsageInfo* method), 126
- `__repr__` () (*fastr.core.tool.Tool* method), 129
- `__repr__` () (*fastr.core.version.Version* method), 132
- `__repr__` () (*fastr.datatypes.BaseDataType* method), 141
- `__repr__` () (*fastr.datatypes.Deferred* method), 147
- `__repr__` () (*fastr.exceptions.FastrError* method), 87
- `__repr__` () (*fastr.exceptions.FastrSerializationError* method), 91
- `__repr__` () (*fastr.execution.environmentmodules.EnvironmentModule* method), 153
- `__repr__` () (*fastr.execution.job.Job* method), 170
- `__repr__` () (*fastr.execution.job.SinkJob* method), 177
- `__repr__` () (*fastr.execution.job.SourceJob* method), 178
- `__repr__` () (*fastr.execution.linkrun.LinkRun* method), 180
- `__repr__` () (*fastr.execution.networkrun.NetworkRun* method), 186
- `__repr__` () (*fastr.execution.noderun.NodeRun* method), 188
- `__repr__` () (*fastr.helpers.configmanager.Config* method), 203
- `__repr__` () (*fastr.helpers.lazy\_module.LazyModule* method), 211
- `__repr__` () (*fastr.planning.inputoutput.BaseInputOutput* method), 223
- `__repr__` () (*fastr.planning.link.Link* method), 238
- `__repr__` () (*fastr.planning.network.Network* method), 241
- `__repr__` () (*fastr.planning.node.Node* method), 249
- `__repr__` () (*fastr.utils.cmd.upgrade.FastrNamespaceType* method), 308
- `__rrshift__` () (*fastr.api.Input* method), 83
- `__rrshift__` () (*fastr.planning.inputoutput.BaseInput* method), 221
- `__setitem__` () (*fastr.core.samples.ContainsSamples* method), 114
- `__setitem__` () (*fastr.core.samples.SampleCollection* method), 118
- `__setitem__` () (*fastr.core.samples.SampleValue* method), 125
- `__setitem__` () (*fastr.execution.inputoutputrun.OutputRun* method), 161
- `__setitem__` () (*fastr.execution.inputoutputrun.SourceOutputRun* method), 163
- `__setitem__` () (*fastr.execution.inputoutputrun.SubOutputRun* method), 167
- `__setitem__` () (*fastr.helpers.configmanager.EmptyDefault* method), 205
- `__setitem__` () (*fastr.planning.inputgroup.InputGroup* method), 217
- `__setitem__` () (*fastr.planning.inputoutput.Input* method), 225
- `__setitem__` () (*fastr.planning.node.InputDict* method), 247
- `__setitem__` () (*fastr.planning.node.OutputDict* method), 251
- `__setitem__` () (*fastr.plugins.managers.pluginmanager.PluginManager* method), 297
- `__setitem__` () (*fastr.plugins.managers.pluginmanager.PluginsView* method), 299
- `__setstate__` () (*fastr.api.ResourceLimit* method), 96
- `__setstate__` () (*fastr.core.interface.Interface* method), 107
- `__setstate__` () (*fastr.core.resourcelimit.ResourceLimit* method), 113
- `__setstate__` () (*fastr.core.samples.SampleValue* method), 125
- `__setstate__` () (*fastr.core.tool.Tool* method), 129
- `__setstate__` () (*fastr.datatypes.BaseDataType* method), 141
- `__setstate__` () (*fastr.datatypes.Deferred* method), 147
- `__setstate__` () (*fastr.execution.inputoutputrun.InputRun* method), 157
- `__setstate__` () (*fastr.execution.inputoutputrun.OutputRun* method), 161
- `__setstate__` () (*fastr.execution.inputoutputrun.SubInputRun* method), 165

[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.inputoutputrun.SubOutputRun* method), 167  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.job.Job* method), 170  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.job.SinkJob* method), 177  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.job.SourceJob* method), 178  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.linkrun.LinkRun* method), 180  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.macronoderun.MacroNodeRun* method), 182  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.networkrun.NetworkRun* method), 186  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.noderun.NodeRun* method), 188  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.sinknoderun.SinkNodeRun* method), 191  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.sourcenoderun.ConstantNodeRun* method), 193  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.execution.sourcenoderun.SourceNodeRun* method), 194  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.inputoutput.BaseInputOutput* method), 223  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.inputoutput.Input* method), 226  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.inputoutput.Output* method), 230  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.inputoutput.SubInput* method), 233  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.inputoutput.SubOutput* method), 235  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.link.Link* method), 238  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.network.Network* method), 241  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.node.ConstantNode* method), 246  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.node.MacroNode* method), 247  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.node.Node* method), 249  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.node.SinkNode* method), 251  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.planning.node.SourceNode* method), 252  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.plugins.FastrInterface* method), 262  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.plugins.FlowInterface* method), 265  
[\\_\\_setstate\\_\\_\(\)](#) (*fastr.plugins.NipyInterface* method), 271  
[\\_\\_sizeof\\_\\_\(\)](#) (in module *fastr.\_\_init\_\_*), 85  
[\\_\\_slots\\_\\_](#) (*fastr.api.ResourceLimit* attribute), 96  
[\\_\\_slots\\_\\_](#) (*fastr.core.dimension.Dimension* attribute), 104  
[\\_\\_slots\\_\\_](#) (*fastr.core.resourcelimit.ResourceLimit* attribute), 113  
[\\_\\_slots\\_\\_](#) (*fastr.core.target.SystemUsageInfo* attribute), 127  
[\\_\\_slots\\_\\_](#) (*fastr.utils.cmd.upgrade.FastrNamespaceType* attribute), 308  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.AnyCardinalitySpec* method), 97  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.AsCardinalitySpec* method), 98  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.CardinalitySpec* method), 99  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.ChoiceCardinalitySpec* method), 100  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.IntCardinalitySpec* method), 101  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.MaxCardinalitySpec* method), 101  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.MinCardinalitySpec* method), 102  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.RangeCardinalitySpec* method), 102  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.cardinality.ValueCardinalitySpec* method), 102  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.samples.SampleBaseId* method), 116  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.samples.SampleIndex* method), 119  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.tool.Tool* method), 129  
[\\_\\_str\\_\\_\(\)](#) (*fastr.core.version.Version* method), 132  
[\\_\\_str\\_\\_\(\)](#) (*fastr.datatypes.BaseDataType* method), 141  
[\\_\\_str\\_\\_\(\)](#) (*fastr.exceptions.FastrError* method), 87  
[\\_\\_str\\_\\_\(\)](#) (*fastr.exceptions.FastrExecutableNotFoundError* method), 88  
[\\_\\_str\\_\\_\(\)](#) (*fastr.exceptions.FastrScriptNotFoundError* method), 91  
[\\_\\_str\\_\\_\(\)](#) (*fastr.exceptions.FastrSerializationError* method), 92  
[\\_\\_str\\_\\_\(\)](#) (*fastr.execution.inputoutputrun.InputRun* method), 157  
[\\_\\_str\\_\\_\(\)](#) (*fastr.execution.inputoutputrun.NamedSubinputRun* method), 160  
[\\_\\_str\\_\\_\(\)](#) (*fastr.execution.inputoutputrun.OutputRun* method), 161  
[\\_\\_str\\_\\_\(\)](#) (*fastr.execution.inputoutputrun.SubInputRun* method), 165  
[\\_\\_str\\_\\_\(\)](#) (*fastr.execution.inputoutputrun.SubOutputRun* method), 167  
[\\_\\_str\\_\\_\(\)](#) (*fastr.execution.noderun.NodeRun* method), 188  
[\\_\\_str\\_\\_\(\)](#) (*fastr.helpers.schematotable.SchemaPrinter* method), 213  
[\\_\\_str\\_\\_\(\)](#) (*fastr.planning.inputoutput.Input* method), 226

---

<code>__str__()</code> ( <i>fastr.planning.inputoutput.NamedSubInput method</i> ), 229	<code>__tree_hash__</code> ( <i>fastr.plugins.managers.toolmanager.ToolManager attribute</i> ), 301
<code>__str__()</code> ( <i>fastr.planning.inputoutput.Output method</i> ), 230	<code>__updatefunc__()</code> ( <i>fastr.planning.inputgroup.InputGroup method</i> ), 217
<code>__str__()</code> ( <i>fastr.planning.inputoutput.SubInput method</i> ), 233	<code>__updatetriggers__</code> ( <i>fastr.planning.inputgroup.InputGroup attribute</i> ), 217
<code>__str__()</code> ( <i>fastr.planning.inputoutput.SubOutput method</i> ), 236	<code>__weakref__</code> ( <i>fastr.core.cardinality.CardinalitySpec attribute</i> ), 99
<code>__str__()</code> ( <i>fastr.planning.node.Node method</i> ), 249	<code>__weakref__</code> ( <i>fastr.core.dimension.HasDimensions attribute</i> ), 105
<code>__subclasshook__()</code> ( <i>fastr.datatypes.DataTypeManager method</i> ), 144	<code>__weakref__</code> ( <i>fastr.core.interface.InterfaceResult attribute</i> ), 108
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.executionpluginmanager.ExecutionPluginManager method</i> ), 291	<code>__weakref__</code> ( <i>fastr.core.provenance.Provenance attribute</i> ), 108
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.interfacemanager.InterfacePluginManager method</i> ), 292	<code>__weakref__</code> ( <i>fastr.core.samples.SampleCollection attribute</i> ), 108
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.iopluginmanager.IOPPluginManager method</i> ), 293	<code>__weakref__</code> ( <i>fastr.core.samples.SampleValue attribute</i> ), 108
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.networkmanager.NetworkManager method</i> ), 294	<code>__weakref__</code> ( <i>fastr.core.target.ProcessUsageCollection attribute</i> ), 108
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.objectmanager.ObjectManager method</i> ), 296	<code>__weakref__</code> ( <i>fastr.core.target.TargetResult attribute</i> ), 108
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.pluginmanager.PluginManager method</i> ), 297	<code>__weakref__</code> ( <i>fastr.core.vfs.VirtualFileSystem attribute</i> ), 108
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.pluginmanager.PluginSubManager method</i> ), 298	<code>__weakref__</code> ( <i>fastr.exceptions.FastrError attribute</i> ), 87
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.targetmanager.TargetManager method</i> ), 300	<code>__weakref__</code> ( <i>fastr.exceptions.FastrIOError attribute</i> ), 88
<code>__subclasshook__()</code> ( <i>fastr.plugins.managers.toolmanager.ToolManager method</i> ), 301	<code>__weakref__</code> ( <i>fastr.exceptions.FastrImportError attribute</i> ), 88
<code>__subclasshook__()</code> ( <i>in module fastr.__init__</i> ), 85	<code>__weakref__</code> ( <i>fastr.exceptions.FastrOSError attribute</i> ), 90
<code>__tree_hash__</code> ( <i>fastr.core.samples.SampleBaseId attribute</i> ), 116	<code>__weakref__</code> ( <i>fastr.execution.environmentmodules.EnvironmentModules attribute</i> ), 153
<code>__tree_hash__</code> ( <i>fastr.core.samples.SampleId attribute</i> ), 119	<code>__weakref__</code> ( <i>fastr.execution.networkanalyzer.NetworkAnalyzer attribute</i> ), 183
<code>__tree_hash__</code> ( <i>fastr.core.samples.SampleIndex attribute</i> ), 119	<code>__weakref__</code> ( <i>fastr.execution.networkchunker.NetworkChunker attribute</i> ), 184
<code>__tree_hash__</code> ( <i>fastr.datatypes.DataTypeManager attribute</i> ), 144	<code>__weakref__</code> ( <i>fastr.helpers.classproperty.ClassPropertyDescriptor attribute</i> ), 205
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.executionpluginmanager.ExecutionPluginManager attribute</i> ), 291	<code>__weakref__</code> ( <i>fastr.helpers.configmanager.Config attribute</i> ), 205
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.interfacemanager.InterfacePluginManager attribute</i> ), 292	<code>__weakref__</code> ( <i>fastr.helpers.configmanager.EmptyDefault attribute</i> ), 205
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.iopluginmanager.IOPPluginManager attribute</i> ), 293	<code>__weakref__</code> ( <i>fastr.helpers.filesynchelper.FileSyncHelper attribute</i> ), 207
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.networkmanager.NetworkManager attribute</i> ), 294	<code>__weakref__</code> ( <i>fastr.helpers.lockfile.DirectoryLock attribute</i> ), 211
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.objectmanager.ObjectManager attribute</i> ), 296	<code>__weakref__</code> ( <i>fastr.helpers.schematotable.SchemaPrinter attribute</i> ), 213
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.pluginmanager.PluginManager attribute</i> ), 297	<code>__weakref__</code> ( <i>fastr.plugins.managers.pluginmanager.PluginsView attribute</i> ), 299
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.pluginmanager.PluginSubManager attribute</i> ), 298	<code>__weakref__</code> ( <i>fastr.utils.cmd.upgrade.dummy_container attribute</i> ), 308
<code>__tree_hash__</code> ( <i>fastr.plugins.managers.targetmanager.TargetManager attribute</i> ), 300	

---

## A

- `abort()` (*fastr.execution.networkrun.NetworkRun* method), 186
- `abstract` (*fastr.core.vfs.VirtualFileSystem* attribute), 135
- `acquire()` (*fastr.helpers.lockfile.DirectoryLock* method), 211
- `action()` (*fastr.datatypes.DataType* method), 143
- `activate()` (*fastr.plugins.ElasticsearchReporter* method), 259
- `activate()` (*fastr.plugins.PimReporter* method), 273
- `activate()` (*fastr.plugins.reportingplugin.ReportingPlugin* method), 290
- `activity()` (*fastr.core.provenance.Provenance* method), 111
- `add_link()` (*fastr.planning.network.Network* method), 241
- `add_node()` (*fastr.planning.network.Network* method), 241
- `add_parser_doc_link()` (in module *fastr.utils.cmd*), 305
- `add_stepid()` (*fastr.planning.network.Network* method), 241
- `AdvancedFlowNode` (class in *fastr.planning.node*), 244
- `AdvancedFlowNodeRun` (class in *fastr.execution.flownoderun*), 154
- `AdvancedFlowOutput` (class in *fastr.planning.inputoutput*), 220
- `AdvancedFlowOutputRun` (class in *fastr.execution.inputoutputrun*), 155
- `agent()` (*fastr.core.provenance.Provenance* method), 111
- `aggregate()` (*fastr.core.target.ProcessUsageCollection* method), 126
- `all` (*fastr.execution.job.JobCleanupLevel* attribute), 174
- `analyze_network()` (*fastr.execution.networkanalyzer.DefaultNetworkAnalyzer* method), 183
- `analyze_network()` (*fastr.execution.networkanalyzer.NetworkAnalyzer* method), 183
- `any_of_draft4()` (in module *fastr.helpers.jsonschemaparser*), 209
- `AnyCardinalitySpec` (class in *fastr.core.cardinality*), 97
- `AnyFile` (class in *fastr.datatypes*), 139
- `AnyType` (class in *fastr.datatypes*), 139
- `append()` (*fastr.api.Input* method), 83
- `append()` (*fastr.core.target.ProcessUsageCollection* method), 126
- `append()` (*fastr.helpers.configmanager.EmptyDefault* method), 205
- `append()` (*fastr.planning.inputoutput.Input* method), 226
- `as_dict()` (*fastr.core.target.TargetResult* method), 128
- `AsCardinalitySpec` (class in *fastr.core.cardinality*), 97
- `asdict()` (*fastr.core.interface.InputSpec* method), 106
- `asdict()` (*fastr.core.interface.OutputSpec* method), 108
- `asdict()` (*fastr.helpers.configmanager.EmptyDefault* method), 205
- `aslist()` (*fastr.helpers.configmanager.EmptyDefault* method), 205
- `authors` (*fastr.core.tool.Tool* attribute), 129
- `automatic` (*fastr.execution.inputoutputrun.OutputRun* property), 161
- `automatic` (*fastr.planning.inputoutput.BaseOutput* property), 224
- `avail()` (*fastr.execution.environmentmodules.EnvironmentModules* method), 153
- `avail_modules` (*fastr.execution.environmentmodules.EnvironmentModule* property), 153

## B

- `BaseDataType` (class in *fastr.datatypes*), 140
- `BaseInput` (class in *fastr.planning.inputoutput*), 221
- `BaseInputGroupCombiner` (class in *fastr.planning.inputgroupcombiner*), 218
- `BaseInputOutput` (class in *fastr.planning.inputoutput*), 222
- `BaseInputRun` (class in *fastr.execution.inputoutputrun*), 155
- `basename()` (in module *fastr.data.url*), 137
- `BaseNode` (class in *fastr.planning.node*), 244
- `BaseNodeRun` (class in *fastr.execution.basenoderun*), 151
- `BaseOutput` (class in *fastr.planning.inputoutput*), 224
- `blocking` (*fastr.execution.flownoderun.FlowNodeRun* property), 155
- `blocking` (*fastr.execution.noderun.NodeRun* property), 188
- `blocking` (*fastr.planning.inputoutput.BaseOutput* property), 224
- `blocking` (*fastr.planning.node.FlowNode* property), 246
- `blocking` (*fastr.planning.node.Node* property), 249
- `BlockingExecution` (class in *fastr.plugins*), 253
- `build` (*fastr.core.version.Version* property), 132

## C

- `calculate_execution_cardinality()` (*fastr.core.cardinality.AsCardinalitySpec* method), 98
- `calculate_execution_cardinality()` (*fastr.core.cardinality.CardinalitySpec* method), 99
- `calculate_execution_cardinality()` (*fastr.core.cardinality.IntCardinalitySpec* method), 101
- `calculate_execution_cardinality()` (*fastr.core.cardinality.ValueCardinalitySpec* method), 102
- `calculate_job_cardinality()` (*fastr.core.cardinality.AsCardinalitySpec* method), 98



- method*), 98
- `calculate_job_cardinality()` (*fastr.core.cardinality.CardinalitySpec method*), 99
- `calculate_job_cardinality()` (*fastr.core.cardinality.IntCardinalitySpec method*), 101
- `calculate_job_cardinality()` (*fastr.core.cardinality.ValueCardinalitySpec method*), 102
- `calculate_planning_cardinality()` (*fastr.core.cardinality.AsCardinalitySpec method*), 98
- `calculate_planning_cardinality()` (*fastr.core.cardinality.CardinalitySpec method*), 100
- `calculate_planning_cardinality()` (*fastr.core.cardinality.IntCardinalitySpec method*), 101
- `call_subprocess()` (*fastr.core.target.SubprocessBasedTarget method*), 126
- `cancel` (*fastr.plugins.executionplugin.JobAction attribute*), 290
- `cancel_job()` (*fastr.plugins.executionplugin.ExecutionPlugin method*), 288
- `cancelled` (*fastr.execution.job.JobState attribute*), 176
- `CANCELS_DEPENDENCIES` (*fastr.plugins.DRMAAExecution attribute*), 256
- `CANCELS_DEPENDENCIES` (*fastr.plugins.executionplugin.ExecutionPlugin attribute*), 287
- `cardinality` (*fastr.core.samples.SampleItemBase property*), 121
- `cardinality()` (*fastr.execution.inputoutputrun.InputRun method*), 157
- `cardinality()` (*fastr.execution.inputoutputrun.OutputRun method*), 161
- `cardinality()` (*fastr.execution.inputoutputrun.SourceOutputRun method*), 163
- `cardinality()` (*fastr.execution.inputoutputrun.SubInputRun method*), 165
- `cardinality()` (*fastr.execution.inputoutputrun.SubOutputRun method*), 167
- `cardinality()` (*fastr.execution.linkrun.LinkRun method*), 181
- `cardinality()` (*fastr.planning.inputoutput.BaseInputOutput method*), 223
- `cardinality()` (*fastr.planning.inputoutput.Input method*), 226
- `cardinality()` (*fastr.planning.inputoutput.Output method*), 231
- `cardinality()` (*fastr.planning.inputoutput.SourceOutput method*), 232
- `cardinality()` (*fastr.planning.inputoutput.SubInput method*), 233
- `cardinality()` (*fastr.planning.inputoutput.SubOutput method*), 236
- `cardinality()` (*fastr.planning.link.Link method*), 238
- `cardinality_from_nargs()` (*in module fastr.utils.cmd.extract\_argparse*), 305
- `CardinalitySpec` (*class in fastr.core.cardinality*), 98
- `cast()` (*fastr.core.samples.SampleValue method*), 125
- `cast_to_type()` (*fastr.execution.job.Job static method*), 170
- `check_cardinality()` (*fastr.planning.inputoutput.BaseInput method*), 221
- `check_cardinality()` (*fastr.planning.inputoutput.BaseInputOutput method*), 223
- `check_finished()` (*fastr.plugins.RQExecution method*), 275
- `check_finished()` (*fastr.plugins.StrongrExecution method*), 280
- `check_id()` (*fastr.execution.networkrun.NetworkRun method*), 186
- `check_id()` (*fastr.planning.network.Network method*), 241
- `check_input_id()` (*fastr.plugins.FastrInterface method*), 262
- `check_job_requirements()` (*fastr.plugins.executionplugin.ExecutionPlugin method*), 288
- `check_job_status()` (*fastr.plugins.executionplugin.ExecutionPlugin method*), 288
- `check_network()` (*in module fastr.utils.cmd.test*), 307
- `check_networks()` (*in module fastr.utils.cmd.test*), 307
- `check_nr_queued_jobs()` (*fastr.plugins.executionplugin.ExecutionPlugin method*), 289
- `check_output_id()` (*fastr.plugins.FastrInterface method*), 257
- `check_threads()` (*fastr.plugins.DRMAAExecution method*), 257
- `check_tool()` (*in module fastr.utils.cmd.test*), 307
- `check_tools()` (*in module fastr.utils.cmd.test*), 307
- `checksum()` (*fastr.datatypes.BaseDataType method*), 141
- `checksum()` (*fastr.datatypes.Deferred method*), 147
- `checksum()` (*fastr.datatypes.URLType method*), 150
- `checksum()` (*in module fastr.helpers.checksum*), 197
- `checksum_directory()` (*in module fastr.helpers.checksum*), 197
- `ChoiceCardinalitySpec` (*class in fastr.core.cardinality*), 100
- `chunk_network()` (*fastr.execution.networkchunker.DefaultNetworkChunker method*), 184

[chunk\\_network\(\)](#) (*fastr.execution.networkchunker.NetworkChunker* method), 178  
[method](#)), 184  
[cite](#) (*fastr.core.tool.Tool* attribute), 129  
[classproperty\(\)](#) (in module *fastr.helpers.classproperty*), 198  
[ClassPropertyDescriptor](#) (class in *fastr.helpers.classproperty*), 198  
[clean\(\)](#) (*fastr.execution.job.Job* method), 170  
[clean\\_free\\_jobs\(\)](#) (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289  
[cleanup\(\)](#) (*fastr.core.ioplugin.IOPlugin* method), 109  
[cleanup\(\)](#) (*fastr.plugins.BlockingExecution* method), 254  
[cleanup\(\)](#) (*fastr.plugins.DRMAAExecution* method), 257  
[cleanup\(\)](#) (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289  
[cleanup\(\)](#) (*fastr.plugins.LinearExecution* method), 267  
[cleanup\(\)](#) (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), 293  
[cleanup\(\)](#) (*fastr.plugins.ProcessPoolExecution* method), 274  
[cleanup\(\)](#) (*fastr.plugins.RQExecution* method), 275  
[cleanup\(\)](#) (*fastr.plugins.S3Filesystem* method), 276  
[cleanup\(\)](#) (*fastr.plugins.SlurmExecution* method), 279  
[cleanup\(\)](#) (*fastr.plugins.StrongrExecution* method), 280  
[cleanup\(\)](#) (*fastr.plugins.XNATStorage* method), 285  
[clear\(\)](#) (*fastr.execution.environmentmodules.EnvironmentModules* method), 153  
[clear\(\)](#) (*fastr.planning.inputoutput.Input* method), 226  
[clear\\_version\(\)](#) (in module *fastr.version*), 94  
[collapse](#) (*fastr.api.Link* property), 82  
[collapse](#) (*fastr.execution.linkrun.LinkRun* property), 181  
[collapse](#) (*fastr.planning.link.Link* property), 238  
[collapse\\_indexes](#) (*fastr.execution.linkrun.LinkRun* property), 181  
[collapse\\_indexes](#) (*fastr.planning.link.Link* property), 239  
[collect\\_errors\(\)](#) (*fastr.plugins.FastrInterface* static method), 262  
[collect\\_input\\_argument\\_provenance\(\)](#) (*fastr.core.provenance.Provenance* method), 111  
[collect\\_jobs\(\)](#) (*fastr.plugins.DRMAAExecution* method), 257  
[collect\\_provenance\(\)](#) (*fastr.core.provenance.Provenance* method), 111  
[collect\\_provenance\(\)](#) (*fastr.execution.job.InlineJob* method), 169  
[collect\\_provenance\(\)](#) (*fastr.execution.job.Job* method), 170  
[collect\\_provenance\(\)](#) (*fastr.execution.job.SourceJob* method), 178  
[collect\\_results\(\)](#) (*fastr.plugins.FastrInterface* method), 262  
[collector\\_plugin\\_type](#) (*fastr.plugins.FastrInterface* attribute), 262  
[collectors](#) (*fastr.plugins.FastrInterface* attribute), 262  
[combine\(\)](#) (*fastr.core.samples.SampleItemBase* static method), 121  
[combine\\_dimensions\(\)](#) (*fastr.core.samples.SampleState* class method), 124  
[combine\\_dimensions\(\)](#) (*fastr.core.dimension.ForwardsDimensions* method), 105  
[command](#) (*fastr.core.tool.Tool* attribute), 129  
[COMMAND\\_DUMP](#) (*fastr.execution.job.Job* attribute), 169  
[command\\_version](#) (*fastr.core.tool.Tool* property), 129  
[commandfile](#) (*fastr.execution.job.Job* property), 170  
[commandurl](#) (*fastr.execution.job.Job* property), 170  
[CommandSeparatedValueFile](#) (class in *fastr.plugins*), 254  
[compare\\_execution\\_dir\(\)](#) (in module *fastr.utils.compare*), 302  
[compare\\_job\\_dirs\(\)](#) (in module *fastr.utils.compare*), 302  
[compare\\_job\\_output\\_data\(\)](#) (in module *fastr.utils.compare*), 302  
[compare\\_output\\_data\(\)](#) (*fastr.core.tool.Tool* static method), 130  
[compare\\_set\(\)](#) (in module *fastr.utils.compare*), 302  
[compare\\_value\\_dict\\_item\(\)](#) (in module *fastr.utils.compare*), 302  
[compare\\_value\\_list\(\)](#) (in module *fastr.utils.compare*), 302  
[Config](#) (class in *fastr.helpers.configmanager*), 199  
[config](#) (in module *fastr.helpers*), 195  
[configuration\\_fields](#) (*fastr.plugins.DRMAAExecution* attribute), 257  
[configuration\\_fields](#) (*fastr.plugins.ElasticsearchReporter* attribute), 259  
[configuration\\_fields](#) (*fastr.plugins.PimReporter* attribute), 273  
[configuration\\_fields](#) (*fastr.plugins.ProcessPoolExecution* attribute), 274  
[configuration\\_fields](#) (*fastr.plugins.RQExecution* attribute), 275  
[configuration\\_fields](#) (*fastr.plugins.SlurmExecution* attribute), 279  
[configuration\\_fields](#) (*fastr.plugins.StrongrExecution* attribute), 280  
[connect\(\)](#) (*fastr.plugins.XNATStorage* method), 285  
[constant\\_id](#) (*fastr.planning.inputoutput.Input* prop-

- erty*), 226
  - `constant_id` (*fastr.planning.inputoutput.NamedSubInput property*), 229
  - `constant_id` (*fastr.planning.inputoutput.SubInput property*), 233
  - `constant_id()` (*fastr.planning.inputoutput.BaseInput method*), 221
  - `constantlist` (*fastr.execution.networkrun.NetworkRun property*), 186
  - `ConstantNode` (*class in fastr.planning.node*), 245
  - `ConstantNodeRun` (*class in fastr.execution.sourcenoderun*), 192
  - `container` (*fastr.plugins.DockerTarget property*), 258
  - `ContainsSamples` (*class in fastr.core.samples*), 114
  - `content()` (*fastr.datatypes.URLType class method*), 150
  - `copy()` (*fastr.api.ResourceLimit method*), 96
  - `copy()` (*fastr.core.dimension.Dimension method*), 104
  - `copy()` (*fastr.core.resourcelimit.ResourceLimit method*), 113
  - `copy_file_dir()` (*fastr.core.vfs.VirtualFileSystem static method*), 135
  - `cores` (*fastr.api.ResourceLimit property*), 96
  - `cores` (*fastr.core.resourcelimit.ResourceLimit property*), 113
  - `cpu_percent` (*fastr.core.target.SystemUsageInfo property*), 127
  - `create_cardinality()` (*in module fastr.core.cardinality*), 103
  - `create_constant()` (*fastr.api.Network method*), 78
  - `create_constant()` (*fastr.planning.network.Network method*), 242
  - `create_enumtype()` (*fastr.datatypes.DataTypeManager method*), 144
  - `create_ioplugin_tool()` (*fastr.plugins.managers.iopluginmanager.IOPuginManager static method*), 293
  - `create_job()` (*fastr.execution.noderun.NodeRun method*), 188
  - `create_job()` (*fastr.execution.sinknoderun.SinkNodeRun method*), 191
  - `create_job()` (*fastr.execution.sourcenoderun.SourceNodeRun method*), 194
  - `create_link()` (*fastr.api.Network method*), 79
  - `create_link()` (*fastr.planning.network.Network method*), 242
  - `create_link_from()` (*fastr.planning.inputoutput.BaseInput method*), 222
  - `create_macro()` (*fastr.api.Network method*), 79
  - `create_macro()` (*fastr.planning.network.Network method*), 242
  - `create_native_spec()` (*fastr.plugins.DRMAAExecution method*), 257
  - `create_network()` (*fastr.api method*), 78
  - `create_network()` (*in module fastr.api*), 97
  - `create_network_copy()` (*fastr.api method*), 78
  - `create_network_copy()` (*in module fastr.api*), 97
  - `create_network_parser()` (*in module fastr.utils.cmd.run*), 306
  - `create_node()` (*fastr.api.Network method*), 79
  - `create_node()` (*fastr.planning.network.Network method*), 242
  - `create_payload()` (*fastr.execution.job.Job method*), 170
  - `create_payload()` (*fastr.execution.job.SinkJob method*), 177
  - `create_payload()` (*fastr.execution.job.SourceJob method*), 178
  - `create_reference()` (*fastr.core.tool.Tool method*), 130
  - `create_reference()` (*fastr.planning.network.Network method*), 243
  - `create_rest_table()` (*in module fastr.helpers.rest\_generation*), 212
  - `create_sink()` (*fastr.api.Network method*), 80
  - `create_sink()` (*fastr.planning.network.Network method*), 243
  - `create_source()` (*fastr.api.Network method*), 80
  - `create_source()` (*fastr.planning.network.Network method*), 243
  - `create_tool_test()` (*in module fastr.utils.verify*), 303
  - `create_vfs_url()` (*in module fastr.data.url*), 137
  - `create_zip()` (*in module fastr.utils.cmd.dump*), 305
  - `created` (*fastr.execution.job.JobState attribute*), 176
  - `createobj()` (*fastr.execution.linkrun.LinkRun class method*), 181
  - `createobj()` (*fastr.execution.noderun.NodeRun class method*), 189
  - `createobj()` (*fastr.planning.link.Link class method*), 249
  - `createobj()` (*fastr.planning.node.Node class method*), 249
  - `CrossValidation` (*class in fastr.plugins*), 255
- ## D
- `data` (*fastr.core.samples.SampleItemBase property*), 122
  - `data` (*fastr.execution.sourcenoderun.ConstantNodeRun property*), 193
  - `data` (*fastr.planning.node.ConstantNode property*), 246
  - `data` (*fastr.plugins.managers.pluginmanager.PluginSubManager property*), 298
  - `data_uri()` (*fastr.core.provenance.Provenance static method*), 111
  - `DataType` (*class in fastr.datatypes*), 142
  - `datatype` (*fastr.execution.inputoutputrun.InputRun property*), 157
  - `datatype` (*fastr.execution.inputoutputrun.OutputRun property*), 162

- ul style="list-style-type: none; padding-left: 0;">
- `datatype` (*fastr.execution.inputoutputrun.SubOutputRun* property), 168
- `datatype` (*fastr.execution.sinknoderun.SinkNodeRun* property), 191
- `datatype` (*fastr.execution.sourcenoderun.SourceNodeRun* property), 194
- `datatype` (*fastr.planning.inputoutput.BaseInputOutput* property), 224
- `datatype` (*fastr.planning.inputoutput.Input* property), 226
- `datatype` (*fastr.planning.inputoutput.Output* property), 231
- `datatype` (*fastr.planning.inputoutput.SubOutput* property), 236
- `datatype` (*fastr.planning.node.SinkNode* property), 252
- `datatype` (*fastr.planning.node.SourceNode* property), 252
- `datatype_from_type()` (in module *fastr.utils.cmd.extract\_argparse*), 305
- `DataTypeManager` (class in *fastr.datatypes*), 143
- `date_version_matcher` (*fastr.core.version.Version* attribute), 132
- `deactivate()` (*fastr.plugins.reportingplugin.ReportingPlugin* method), 290
- `debug` (*fastr.helpers.configmanager.Config* property), 203
- `default` (*fastr.planning.inputoutput.BaseInput* property), 222
- `DEFAULT_FIELDS` (*fastr.helpers.configmanager.Config* attribute), 199
- `DEFAULT_TARGET_CLASS` (*fastr.core.tool.Tool* attribute), 128
- `DefaultInputGroupCombiner` (class in *fastr.planning.inputgroupcombiner*), 219
- `DefaultNetworkAnalyzer` (class in *fastr.execution.networkanalyzer*), 183
- `DefaultNetworkChunker` (class in *fastr.execution.networkchunker*), 184
- `Deferred` (class in *fastr.datatypes*), 146
- `dependencies()` (*fastr.planning.network.Network* method), 243
- `descend()` (*fastr.helpers.schematotable.SchemaPrinter* method), 213
- `description` (*fastr.core.tool.Tool* attribute), 130
- `description` (*fastr.datatypes.AnyFile* attribute), 139
- `description` (*fastr.datatypes.AnyType* attribute), 140
- `description` (*fastr.datatypes.BaseDataType* attribute), 141
- `description` (*fastr.datatypes.EnumType* attribute), 148
- `description` (*fastr.execution.inputoutputrun.SubInputRun* property), 165
- `description` (*fastr.planning.inputoutput.BaseInputOutput* property), 224
- `description` (*fastr.planning.inputoutput.SubInput* property), 233
- `description_type` (*fastr.planning.inputoutput.BaseInput* attribute), 222
- `description_type` (*fastr.planning.inputoutput.BaseInputOutput* attribute), 224
- `description_type` (*fastr.planning.inputoutput.BaseOutput* attribute), 224
- `deserialize()` (*fastr.datatypes.DataType* class method), 143
- `destroy()` (*fastr.execution.linkrun.LinkRun* method), 181
- `destroy()` (*fastr.planning.link.Link* method), 239
- `dicteq()` (in module *fastr.utils.dicteq*), 302
- `diffdict()` (in module *fastr.utils.dicteq*), 302
- `diffobj()` (in module *fastr.utils.dicteq*), 303
- `diffobj_str()` (in module *fastr.utils.dicteq*), 303
- `Dimension` (class in *fastr.core.dimension*), 104
- `dimensionality` (*fastr.core.samples.SampleItemBase* property), 122
- `dimensions` (*fastr.core.dimension.ForwardsDimensions* property), 105
- `dimensions` (*fastr.core.dimension.HasDimensions* property), 105
- `dimensions` (*fastr.core.samples.ContainsSamples* property), 114
- `dimensions` (*fastr.core.samples.SampleCollection* property), 118
- `dimensions` (*fastr.execution.inputoutputrun.InputRun* property), 157
- `dimensions` (*fastr.execution.inputoutputrun.MacroOutputRun* property), 159
- `dimensions` (*fastr.execution.inputoutputrun.SourceOutputRun* property), 164
- `dimensions` (*fastr.execution.inputoutputrun.SubInputRun* property), 165
- `dimensions` (*fastr.execution.linkrun.LinkRun* property), 181
- `dimensions` (*fastr.planning.inputgroup.InputGroup* property), 217
- `dimensions` (*fastr.planning.inputgroupcombiner.BaseInputGroupCombiner* property), 218
- `dimensions` (*fastr.planning.inputoutput.AdvancedFlowOutput* property), 221
- `dimensions` (*fastr.planning.inputoutput.Input* property), 227
- `dimensions` (*fastr.planning.inputoutput.MacroOutput* property), 228
- `dimensions` (*fastr.planning.inputoutput.Output* property), 231
- `dimensions` (*fastr.planning.inputoutput.SubInput* property), 233
- `dimensions` (*fastr.planning.link.Link* property), 239
- `dimensions` (*fastr.planning.node.FlowNode* property), 246
- `dimensions` (*fastr.planning.node.Node* property), 249



- dimensions (*fastr.planning.node.SourceNode* property), 253  
 dimnames (*fastr.core.dimension.HasDimensions* property), 106  
 dimnames (*fastr.execution.flownoderun.FlowNodeRun* property), 155  
 dimnames (*fastr.execution.noderun.NodeRun* property), 189  
 dimnames (*fastr.execution.sourcenoderun.SourceNodeRun* property), 194  
 dimnames (*fastr.planning.node.Node* property), 250  
 dir\_list() (in module *fastr.helpers.clear\_pycs*), 199  
 directory() (in module *fastr.utils.cmd.test*), 307  
 DirectoryLock (class in *fastr.helpers.lockfile*), 211  
 dirname() (in module *fastr.data.url*), 137  
 dirurl() (in module *fastr.data.url*), 137  
 dispatch\_callbacks() (*fastr.plugins.DRMAAExecution* method), 257  
 DockerTarget (class in *fastr.plugins*), 258  
 done (*fastr.execution.job.JobState* property), 176  
 dot\_extension (*fastr.datatypes.BaseDataType* attribute), 142  
 draw() (*fastr.api.Network* method), 80  
 draw() (*fastr.planning.link.Link* method), 239  
 draw() (*fastr.planning.network.Network* method), 243  
 draw() (*fastr.planning.node.ConstantNode* method), 246  
 draw() (*fastr.planning.node.MacroNode* method), 248  
 draw() (*fastr.planning.node.Node* method), 250  
 draw() (*fastr.planning.node.SinkNode* method), 252  
 draw() (*fastr.planning.node.SourceNode* method), 253  
 draw\_id() (*fastr.planning.node.Node* method), 250  
 draw\_link\_target() (*fastr.planning.node.MacroNode* method), 248  
 draw\_link\_target() (*fastr.planning.node.Node* method), 250  
 draw\_network() (*fastr.planning.network.Network* method), 243  
 DRMAAExecution (class in *fastr.plugins*), 256  
 dummy\_container (class in *fastr.utils.cmd.upgrade*), 308  
 dump() (in module *fastr.helpers.xmltodict*), 215  
 dumps() (in module *fastr.helpers.xmltodict*), 215  
 DYNAMIC\_LIBRARY\_PATH\_DICT (*fastr.plugins.LocalBinaryTarget* attribute), 269  
**E**  
 elasticsearch\_update\_status() (*fastr.plugins.ElasticsearchReporter* method), 259  
 ElasticsearchReporter (class in *fastr.plugins*), 259  
 emit() (*fastr.helpers.events.FastrLogEventHandler* method), 206  
 emit\_event() (in module *fastr.helpers.events*), 206  
 empty (*fastr.planning.inputgroup.InputGroup* property), 217  
 EmptyDefault (class in *fastr.helpers.configmanager*), 204  
 ensure\_threads() (*fastr.plugins.DRMAAExecution* method), 257  
 ensure\_tmp\_dir() (*fastr.execution.job.Job* method), 170  
 entity() (*fastr.core.provenance.Provenance* method), 111  
 EnumType (class in *fastr.datatypes*), 148  
 EnvironmentModules (class in *fastr.execution.environmentmodules*), 152  
 envmod (*fastr.execution.environmentmodules.ModuleSystem* attribute), 154  
 EventType (class in *fastr.helpers.events*), 206  
 examplesdir (*fastr.helpers.configmanager.Config* property), 203  
 excerpt() (*fastr.exceptions.FastrError* method), 87  
 exec\_worker() (*fastr.plugins.LinearExecution* method), 267  
 execute() (*fastr.api.Network* method), 80  
 execute() (*fastr.core.interface.Interface* method), 107  
 execute() (*fastr.core.tool.Tool* method), 130  
 execute() (*fastr.execution.job.Job* method), 170  
 execute() (*fastr.execution.macronoderun.MacroNodeRun* method), 183  
 execute() (*fastr.execution.networkrun.NetworkRun* method), 186  
 execute() (*fastr.execution.noderun.NodeRun* method), 189  
 execute() (*fastr.planning.network.Network* method), 243  
 execute() (*fastr.plugins.CrossValidation* static method), 255  
 execute() (*fastr.plugins.FastrInterface* method), 262  
 execute() (*fastr.plugins.FlowInterface* method), 265  
 execute() (*fastr.plugins.NipyInterface* method), 271  
 execute\_job() (in module *fastr.execution.executionscript*), 154  
 execution\_done (*fastr.execution.job.JobState* attribute), 176  
 execution\_failed (*fastr.execution.job.JobState* attribute), 176  
 execution\_finished() (*fastr.execution.networkrun.NetworkRun* method), 186  
 execution\_plugin (*fastr.helpers.configmanager.Config* property), 203  
 execution\_skipped (*fastr.execution.job.JobState* attribute), 176  
 ExecutionPlugin (class in *fastr.plugins.executionplugin*), 287  
 ExecutionPluginManager (class in

*fastr.plugins.managers.executionpluginmanager*), *fastr.core.cardinality*  
290 module, 97

*executionscript* (*fastr.helpers.configmanager.Config*  
*property*), 203 *fastr.core.dimension*  
module, 104

*expand* (*fastr.api.Link* *property*), 82 *fastr.core.interface*  
module, 106

*expand* (*fastr.execution.linkrun.LinkRun* *property*), 182 *fastr.core.ioplugin*  
module, 108

*expand* (*fastr.planning.link.Link* *property*), 239 *fastr.core.provenance*  
module, 111

*expand()* (*fastr.core.samples.SampleIndex* *method*), 120 *fastr.core.resourcelimit*  
module, 112

*expand\_network\_scope()*  
(*fastr.core.vfs.VirtualFileSystem* *method*),  
135 *fastr.core.samples*  
module, 114

*expand\_url()* (*fastr.core.ioplugin.IOPlugin* *method*),  
109 *fastr.core.target*  
module, 125

*expand\_url()* (*fastr.core.vfs.VirtualFileSystem* *method*),  
135 *fastr.core.test*  
module, 126

*expand\_url()* (*fastr.plugins.CommaSeperatedValueFile*  
*method*), 255 *fastr.core.tool*  
module, 128

*expand\_url()* (*fastr.plugins.managers.iopluginmanager.IOPlugin* *method*), 293 *fastr.core.version*  
module, 131

*expand\_url()* (*fastr.plugins.S3Filesystem* *method*), 276 *fastr.core.vfs*  
module, 133

*expand\_url()* (*fastr.plugins.VirtualFileSystemRegularExpression* *method*), 282 *fastr.data*  
module, 137

*expand\_url()* (*fastr.plugins.VirtualFileSystemValueList* *method*), 283 *fastr.data.url*  
module, 137

*expand\_url()* (*fastr.plugins.XNATStorage* *method*), 286 *fastr.datatypes*  
module, 139

*expanding* (*fastr.core.interface.Interface* *property*), 107 *fastr.exceptions*  
module, 86

*expanding* (*fastr.plugins.FastrInterface* *property*), 263 *fastr.execution*  
module, 151

*expanding* (*fastr.plugins.FlowInterface* *property*), 266 *fastr.execution.basenoderun*  
module, 151

*expanding* (*fastr.plugins.NipypeInterface* *property*), 272 *fastr.execution.environmentmodules*  
module, 152

*extend()* (*fastr.helpers.configmanager.EmptyDefault*  
*method*), 205 *fastr.execution.executionscript*  
module, 154

*extend()* (in module *fastr.helpers.jsonscemaparser*),  
209 *fastr.execution.flownoderun*  
module, 154

*extension* (*fastr.datatypes.BaseDataType* *attribute*), 142 *fastr.execution.inputoutputrun*  
module, 155

*extra* (*fastr.core.version.Version* *property*), 133 *fastr.execution.job*  
module, 168

*extra\_config\_dirs* (*fastr.helpers.configmanager.Config*  
*property*), 203 *fastr.execution.linkrun*  
module, 179

*extra\_string* (*fastr.core.version.Version* *property*), 133 *fastr.execution.macronoderun*  
module, 182

*extract\_argparser()* (in module *fastr.utils.cmd.extract\_argparse*), 305 *fastr.execution.networkanalyzer*  
module, 183

*extrainfofile* (*fastr.execution.job.Job* *property*), 171 *fastr.execution.networkchunker*  
module, 184

*extrainfourl* (*fastr.execution.job.Job* *property*), 171

## F

FAILED (*fastr.core.samples.SampleState* *attribute*), 123

failed (*fastr.execution.job.JobState* *attribute*), 176

failed\_annotations (*fastr.core.samples.SampleItemBase*  
*property*), 122

*fastr.\_\_init\_\_*  
module, 85

*fastr.api*  
module, 95

*fastr.core*  
module, 97

`fastr.execution.networkrun`  
    module, 185

`fastr.execution.noderun`  
    module, 187

`fastr.execution.sinknoderun`  
    module, 190

`fastr.execution.sourcenoderun`  
    module, 192

`fastr.globals`  
    module, 94

`fastr.helpers`  
    module, 195

`fastr.helpers.checksum`  
    module, 197

`fastr.helpers.classproperty`  
    module, 198

`fastr.helpers.clear_pycs`  
    module, 199

`fastr.helpers.configmanager`  
    module, 199

`fastr.helpers.events`  
    module, 206

`fastr.helpers.filesynchelper`  
    module, 207

`fastr.helpers.iohelpers`  
    module, 208

`fastr.helpers.jsonschemaparser`  
    module, 208

`fastr.helpers.lazy_module`  
    module, 210

`fastr.helpers.lockfile`  
    module, 211

`fastr.helpers.procutils`  
    module, 212

`fastr.helpers.report`  
    module, 212

`fastr.helpers.rest_generation`  
    module, 212

`fastr.helpers.schematotable`  
    module, 212

`fastr.helpers.shellescape`  
    module, 213

`fastr.helpers.sysinfo`  
    module, 214

`fastr.helpers.xmltodict`  
    module, 215

`fastr.planning`  
    module, 216

`fastr.planning.inputgroup`  
    module, 216

`fastr.planning.inputgroupcombiner`  
    module, 218

`fastr.planning.inputoutput`  
    module, 220

`fastr.planning.link`  
    module, 237

`fastr.planning.network`  
    module, 240

`fastr.planning.node`  
    module, 244

`fastr.plugins`  
    module, 253

`fastr.plugins.executionplugin`  
    module, 287

`fastr.plugins.managers`  
    module, 290

`fastr.plugins.managers.executionpluginmanager`  
    module, 290

`fastr.plugins.managers.interfacemanager`  
    module, 291

`fastr.plugins.managers.iopluginmanager`  
    module, 292

`fastr.plugins.managers.networkmanager`  
    module, 294

`fastr.plugins.managers.objectmanager`  
    module, 295

`fastr.plugins.managers.pluginmanager`  
    module, 297

`fastr.plugins.managers.targetmanager`  
    module, 300

`fastr.plugins.managers.toolmanager`  
    module, 300

`fastr.plugins.reportingplugin`  
    module, 290

`fastr.test`  
    module, 301

`fastr.utils`  
    module, 301

`fastr.utils.cmd`  
    module, 305

`fastr.utils.cmd.cat`  
    module, 305

`fastr.utils.cmd.dump`  
    module, 305

`fastr.utils.cmd.execute`  
    module, 305

`fastr.utils.cmd.extract_argparse`  
    module, 305

`fastr.utils.cmd.provenance`  
    module, 306

`fastr.utils.cmd.pylint`  
    module, 306

`fastr.utils.cmd.report`  
    module, 306

`fastr.utils.cmd.run`  
    module, 306

`fastr.utils.cmd.sink`  
    module, 307

`fastr.utils.cmd.source`  
    module, 307

`fastr.utils.cmd.test`  
    module, 307

`fastr.utils.cmd.trace`  
    module, 307

`fastr.utils.cmd.upgrade`  
    module, 308

`fastr.utils.cmd.verify`  
    module, 309

`fastr.utils.compare`  
    module, 302

`fastr.utils.dicteq`  
    module, 302

`fastr.utils.gettools`  
    module, 303

`fastr.utils.multiprocesswrapper`  
    module, 303

`fastr.utils.verify`  
    module, 303

`fastr.version`  
    module, 94

`fastr_cat()` (in module `fastr.utils.cmd.cat`), 305

`fastr_isinstance()` (in module `fastr.datatypes`), 151

`FastrAttributeError`, 86

`FastrCannotChangeAttributeError`, 86

`FastrCardinalityError`, 86

`FastrCollectorError`, 86

`FastrDataTypeFileNotReadable`, 86

`FastrDataTypeMismatchError`, 86

`FastrDataTypeNotAvailableError`, 86

`FastrDataTypeNotInstantiableError`, 86

`FastrDataTypeValueError`, 87

`FastrError`, 87

`FastrErrorInSubprocess`, 87

`FastrExecutableNotFoundError`, 87

`FastrExecutionError`, 88

`FastrFileNotFound`, 88

`FastrImportError`, 88

`FastrIndexError`, 88

`FastrIndexNonexistent`, 88

`FastrInterface` (class in `fastr.plugins`), 260

`FastrIOError`, 88

`FastrKeyError`, 88

`FastrLockNotAcquired`, 89

`FastrLogEventHandler` (class in `fastr.helpers.events`), 206

`FastrLogRecordFilter` (class in `fastr.helpers.configmanager`), 205

`FastrLookupError`, 89

`FastrMountUnknownError`, 89

`FastrNamespaceType` (class in `fastr.utils.cmd.upgrade`), 308

`FastrNetworkMismatchError`, 89

`FastrNetworkUnknownError`, 89

`FastrNodeAlreadyPreparedError`, 89

`FastrNodeNotPreparedError`, 89

`FastrNodeNotValidError`, 90

`FastrNotExecutableError`, 90

`FastrNotImplementedError`, 90

`FastrNoValidTargetError`, 89

`FastrObjectUnknownError`, 90

`FastrOptionalModuleNotAvailableError`, 90

`FastrOSError`, 90

`FastrOutputValidationError`, 90

`FastrParentMismatchError`, 90

`FastrPluginCapabilityNotImplemented`, 90

`FastrPluginNotAvailable`, 91

`FastrPluginNotLoaded`, 91

`FastrRefResolver` (class in `fastr.helpers.jsonschemaparser`), 208

`FastrResultFileNotFound`, 91

`FastrScriptNotFoundError`, 91

`FastrSerializationError`, 91

`FastrSerializationIgnoreDefaultError`, 92

`FastrSerializationInvalidDataError`, 92

`FastrSerializationMethodError`, 92

`FastrSinkDataUnavailableError`, 92

`FastrSizeInvalidError`, 92

`FastrSizeMismatchError`, 92

`FastrSizeUnknownError`, 92

`FastrSourceDataUnavailableError`, 92

`FastrStateError`, 93

`FastrSubprocessNotFinished`, 93

`FastrToolNotAvailableError`, 93

`FastrToolTargetNotFound`, 93

`FastrToolUnknownError`, 93

`FastrToolVersionError`, 93

`FastrTypeError`, 93

`FastrUnknownURLSchemeError`, 93

`FastrValueError`, 93

`FastrVersionInvalidError`, 94

`FastrVersionMismatchError`, 94

`fetch_url()` (`fastr.core.ioplugin.IOPlugin` method), 109

`fetch_url()` (`fastr.core.vfs.VirtualFileSystem` method), 135

`fetch_url()` (`fastr.plugins.FileSystem` method), 264

`fetch_url()` (`fastr.plugins.HTTPPlugin` method), 267

`fetch_url()` (`fastr.plugins.S3Filesystem` method), 277

`fetch_url()` (`fastr.plugins.XNATStorage` method), 286

`fetch_value()` (`fastr.core.ioplugin.IOPlugin` method), 109

`fetch_value()` (`fastr.core.vfs.VirtualFileSystem` method), 135

`fetch_value()` (`fastr.plugins.FileSystem` method), 264

`fetch_value()` (`fastr.plugins.S3Filesystem` method), 277

`filename` (`fastr.datatypes.BaseDataType` attribute), 142



- filename (*fastr.plugins.BlockingExecution* attribute), 254
- filename (*fastr.plugins.CommaSeperatedValueFile* attribute), 255
- filename (*fastr.plugins.CrossValidation* attribute), 256
- filename (*fastr.plugins.DockerTarget* attribute), 258
- filename (*fastr.plugins.DRMAAExecution* attribute), 258
- filename (*fastr.plugins.ElasticsearchReporter* attribute), 259
- filename (*fastr.plugins.FastrInterface* attribute), 263
- filename (*fastr.plugins.FileSystem* attribute), 264
- filename (*fastr.plugins.FlowInterface* attribute), 266
- filename (*fastr.plugins.HTTPPlugin* attribute), 267
- filename (*fastr.plugins.LinearExecution* attribute), 267
- filename (*fastr.plugins.LocalBinaryTarget* attribute), 269
- filename (*fastr.plugins.MacroTarget* attribute), 270
- filename (*fastr.plugins.NetworkScope* attribute), 270
- filename (*fastr.plugins.NipypeInterface* attribute), 272
- filename (*fastr.plugins.Null* attribute), 272
- filename (*fastr.plugins.PimReporter* attribute), 273
- filename (*fastr.plugins.ProcessPoolExecution* attribute), 274
- filename (*fastr.plugins.Reference* attribute), 276
- filename (*fastr.plugins.RQExecution* attribute), 275
- filename (*fastr.plugins.S3Filesystem* attribute), 277
- filename (*fastr.plugins.SimpleReport* attribute), 278
- filename (*fastr.plugins.SingularityTarget* attribute), 278
- filename (*fastr.plugins.SlurmExecution* attribute), 279
- filename (*fastr.plugins.StrongrExecution* attribute), 280
- filename (*fastr.plugins.VirtualFileSystem* attribute), 281
- filename (*fastr.plugins.VirtualFileSystemRegularExpression* attribute), 282
- filename (*fastr.plugins.VirtualFileSystemValueList* attribute), 283
- filename (*fastr.plugins.XNATStorage* attribute), 286
- FileSyncHelper (class in *fastr.helpers.filesynchelper*), 207
- filesynchelper\_enabled() (in module *fastr.helpers.filesynchelper*), 207
- filesynchelper\_url (*fastr.helpers.configmanager.Config* property), 203
- FileSystem (class in *fastr.plugins*), 263
- fill\_output\_argument() (*fastr.execution.job.Job* class method), 171
- filter() (*fastr.helpers.configmanager.FastrLogRecordFilter* method), 205
- filter\_plugin() (*fastr.plugins.managers.pluginmanager.PluginManager* method), 299
- find\_argparser() (in module *fastr.utils.cmd.extract\_argparse*), 305
- find\_commands() (in module *fastr.utils.cmd*), 305
- find\_source\_index() (*fastr.execution.noderun.NodeRun* method), 189
- find\_source\_index() (*fastr.planning.inputgroup.InputGroup* method), 217
- find\_source\_index() (*fastr.planning.node.Node* method), 250
- find\_tool() (in module *fastr.utils.cmd.upgrade*), 308
- finished (*fastr.execution.job.JobState* attribute), 176
- flow\_plugin\_type (*fastr.plugins.FlowInterface* attribute), 266
- flow\_plugins (*fastr.plugins.FlowInterface* attribute), 266
- FlowInterface (class in *fastr.plugins*), 265
- FlowNode (class in *fastr.planning.node*), 246
- FlowNodeRun (class in *fastr.execution.flownoderun*), 155
- ForwardsDimensions (class in *fastr.core.dimension*), 104
- from\_schema() (*fastr.helpers.jsonschemaparser.FastrRefResolver* class method), 208
- full\_split() (in module *fastr.data.url*), 138
- fullid (*fastr.core.samples.SampleCollection* property), 118
- fullid (*fastr.core.tool.Tool* property), 130
- fullid (*fastr.datatypes.BaseDataType* attribute), 142
- fullid (*fastr.datatypes.DataTypeManager* property), 144
- fullid (*fastr.execution.inputoutputrun.InputRun* property), 157
- fullid (*fastr.execution.inputoutputrun.NamedSubInputRun* property), 160
- fullid (*fastr.execution.inputoutputrun.OutputRun* property), 162
- fullid (*fastr.execution.inputoutputrun.SubInputRun* property), 165
- fullid (*fastr.execution.inputoutputrun.SubOutputRun* property), 168
- fullid (*fastr.execution.job.Job* property), 171
- fullid (*fastr.execution.linkrun.LinkRun* property), 182
- fullid (*fastr.execution.networkrun.NetworkRun* property), 186
- fullid (*fastr.execution.noderun.NodeRun* property), 189
- fullid (*fastr.planning.inputgroup.InputGroup* property), 217
- fullid (*fastr.planning.inputgroupcombiner.BaseInputGroupCombiner* property), 218
- fullid (*fastr.planning.inputoutput.BaseInputOutput* property), 224
- fullid (*fastr.planning.inputoutput.Input* property), 227
- fullid (*fastr.planning.inputoutput.NamedSubInput* property), 229
- fullid (*fastr.planning.inputoutput.Output* property), 231
- fullid (*fastr.planning.inputoutput.SubInput* property), 233

`fullid` (*fastr.planning.inputoutput.SubOutput* property), 236

`fullid` (*fastr.planning.link.Link* property), 239

`fullid` (*fastr.planning.network.Network* property), 244

`fullid` (*fastr.planning.node.Node* property), 250

`function_wrapper()` (in module *fastr.utils.multiprocesswrapper*), 303

## G

`GE_NATIVE_SPEC` (*fastr.plugins.DRMAAExecution* attribute), 256

`generate_jobs()` (*fastr.execution.networkrun.NetworkRun* method), 187

`get_arguments()` (*fastr.plugins.FastrInterface* method), 263

`get_base_version()` (in module *fastr.version*), 94

`get_command()` (*fastr.plugins.FastrInterface* method), 263

`get_command_module()` (in module *fastr.utils.cmd*), 305

`get_cpu_usage()` (in module *fastr.helpers.sysinfo*), 214

`get_current_run()` (in module *fastr.globals*), 94

`get_deferred()` (*fastr.execution.job.Job* method), 171

`get_drmaa_info()` (in module *fastr.helpers.sysinfo*), 214

`get_field()` (*fastr.helpers.configmanager.Config* method), 203

`get_git_info()` (in module *fastr.version*), 95

`get_hostinfo()` (in module *fastr.helpers.sysinfo*), 214

`get_job()` (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289

`get_memory_usage()` (in module *fastr.helpers.sysinfo*), 214

`get_message()` (in module *fastr.exceptions*), 94

`get_mounts()` (in module *fastr.helpers.sysinfo*), 214

`get_object_version()` (*fastr.plugins.managers.networkmanager.NetworkManager* method), 294

`get_object_version()` (*fastr.plugins.managers.objectmanager.ObjectManager* method), 296

`get_object_version()` (*fastr.plugins.managers.toolmanager.ToolManager* method), 301

`get_orderreddict_cardinality()` (*fastr.core.cardinality.AsCardinalitySpec* method), 98

`get_os()` (in module *fastr.helpers.sysinfo*), 214

`get_output_datatype()` (*fastr.execution.job.Job* method), 171

`get_output_datatype()` (*fastr.execution.job.SourceJob* method), 178

`get_output_info()` (*fastr.execution.macronoderun.MacroNodeRun* method), 183

`get_output_info()` (*fastr.planning.node.MacroNode* method), 248

`get_parent_provenance()` (*fastr.core.provenance.Provenance* static method), 112

`get_parser()` (in module *fastr.utils.cmd.cat*), 305

`get_parser()` (in module *fastr.utils.cmd.dump*), 305

`get_parser()` (in module *fastr.utils.cmd.execute*), 305

`get_parser()` (in module *fastr.utils.cmd.extract\_argparse*), 306

`get_parser()` (in module *fastr.utils.cmd.provenance*), 306

`get_parser()` (in module *fastr.utils.cmd.pylint*), 306

`get_parser()` (in module *fastr.utils.cmd.report*), 306

`get_parser()` (in module *fastr.utils.cmd.run*), 306

`get_parser()` (in module *fastr.utils.cmd.sink*), 307

`get_parser()` (in module *fastr.utils.cmd.source*), 307

`get_parser()` (in module *fastr.utils.cmd.test*), 307

`get_parser()` (in module *fastr.utils.cmd.trace*), 307

`get_parser()` (in module *fastr.utils.cmd.upgrade*), 308

`get_parser()` (in module *fastr.utils.cmd.verify*), 309

`get_path_from_url()` (in module *fastr.data.url*), 138

`get_pid()` (*fastr.helpers.lockfile.DirectoryLock* method), 211

`get_processes()` (in module *fastr.helpers.sysinfo*), 214

`get_prov_document()` (in module *fastr.utils.cmd.provenance*), 306

`get_python()` (in module *fastr.helpers.sysinfo*), 215

`get_result()` (*fastr.execution.job.InlineJob* method), 169

`get_result()` (*fastr.execution.job.Job* method), 171

`get_result()` (*fastr.execution.job.SinkJob* method), 177

`get_saved_version()` (in module *fastr.version*), 95

`get_sourced_nodes()` (*fastr.execution.inputoutputrun.InputRun* method), 157

`get_sourced_nodes()` (*fastr.execution.inputoutputrun.SubInputRun* method), 165

`get_sourced_nodes()` (*fastr.execution.noderun.NodeRun* method), 189

`get_sourced_nodes()` (*fastr.planning.inputoutput.Input* method), 227

`get_sourced_nodes()` (*fastr.planning.inputoutput.SubInput* method), 234

`get_sourced_nodes()` (*fastr.planning.node.Node* method), 250

`get_sourced_outputs()` (*fastr.execution.inputoutputrun.InputRun* method), 158

`get_sourced_outputs()`

- (*fastr.execution.inputoutputrun.SubInputRun* method), 165
- `get_sourced_outputs()` (*fastr.planning.inputoutput.Input* method), 227
- `get_sourced_outputs()` (*fastr.planning.inputoutput.SubInput* method), 234
- `get_specials()` (*fastr.plugins.FastrInterface* method), 263
- `get_status()` (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289
- `get_subinput_cardinality()` (*fastr.execution.inputoutputrun.InputRun* method), 158
- `get_sysinfo()` (in module *fastr.helpers.sysinfo*), 215
- `get_target()` (*fastr.core.cardinality.AsCardinalitySpec* method), 98
- `get_type()` (*fastr.datatypes.DataTypeManager* method), 145
- `get_type()` (*fastr.plugins.NipypeInterface* method), 272
- `get_url_scheme()` (in module *fastr.data.url*), 138
- `get_users()` (in module *fastr.helpers.sysinfo*), 215
- `get_value()` (*fastr.execution.job.Job* class method), 172
- `getblueprinter()` (in module *fastr.helpers.jsonschemaparser*), 209
- `global_id` (*fastr.execution.networkrun.NetworkRun* property), 187
- `global_id` (*fastr.execution.noderun.NodeRun* property), 189
- `global_id` (*fastr.planning.network.Network* property), 244
- `global_id` (*fastr.planning.node.Node* property), 250
- `gpus` (*fastr.api.ResourceLimit* property), 96
- `gpus` (*fastr.core.resourcelimit.ResourceLimit* property), 113
- `guess_type()` (*fastr.datatypes.DataTypeManager* method), 145
- ## H
- `has_file_promise()` (*fastr.helpers.filesynchelper.FileSynchelper* method), 207
- `has_type()` (*fastr.datatypes.DataTypeManager* method), 145
- `HasDimensions` (class in *fastr.core.dimension*), 105
- `hash` (*fastr.core.tool.Tool* property), 130
- `hash_inputs()` (*fastr.execution.job.Job* method), 172
- `hash_inputs()` (*fastr.execution.job.SinkJob* method), 177
- `hash_inputs()` (*fastr.execution.job.SourceJob* method), 179
- `hash_results()` (*fastr.execution.job.Job* method), 172
- `hashsum()` (in module *fastr.helpers.checksum*), 197
- `HasSamples` (class in *fastr.core.samples*), 114
- `help` (*fastr.core.tool.Tool* attribute), 130
- `hold` (*fastr.execution.job.JobState* attribute), 176
- `hold` (*fastr.plugins.executionplugin.JobAction* attribute), 290
- `hold_job()` (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289
- `HTTPPlugin` (class in *fastr.plugins*), 266
- ## I
- `id` (*fastr.api.Input* property), 83
- `id` (*fastr.api.Link* property), 82
- `id` (*fastr.api.Network* property), 81
- `id` (*fastr.api.Node* property), 82
- `id` (*fastr.api.Output* property), 84
- `id` (*fastr.core.samples.SampleItemBase* property), 122
- `id` (*fastr.core.tool.Tool* property), 130
- `id` (*fastr.datatypes.BaseDataType* attribute), 142
- `id` (*fastr.execution.job.Job* property), 172
- `id` (*fastr.execution.job.SinkJob* property), 177
- `id` (*fastr.execution.networkrun.NetworkRun* property), 187
- `id` (*fastr.execution.noderun.NodeRun* property), 189
- `id` (*fastr.planning.inputoutput.BaseInputOutput* property), 224
- `id` (*fastr.planning.inputoutput.Input* property), 227
- `id` (*fastr.planning.network.Network* property), 244
- `id` (*fastr.planning.node.Node* property), 250
- `idle` (*fastr.execution.job.JobState* property), 176
- `ids()` (*fastr.core.samples.HasSamples* method), 114
- `in_progress` (*fastr.execution.job.JobState* property), 176
- `index` (*fastr.core.samples.SampleItemBase* property), 122
- `index()` (*fastr.execution.inputoutputrun.InputRun* method), 158
- `index()` (*fastr.planning.inputoutput.Input* method), 227
- `indexes()` (*fastr.core.samples.HasSamples* method), 114
- `indexrep` (*fastr.execution.inputoutputrun.SubOutputRun* property), 168
- `indexrep` (*fastr.planning.inputoutput.SubOutput* property), 236
- `INFO_DUMP` (*fastr.execution.job.Job* attribute), 169
- `init_provenance()` (*fastr.core.provenance.Provenance* method), 112
- `InlineJob` (class in *fastr.execution.job*), 168
- `Input` (class in *fastr.api*), 82
- `Input` (class in *fastr.planning.inputoutput*), 224
- `input` (*fastr.api.Node* property), 82
- `input` (*fastr.execution.sinknoderun.SinkNodeRun* property), 191
- `input` (*fastr.planning.node.SinkNode* property), 252
- `input_group` (*fastr.api.Input* property), 84

(*fastr.execution.inputoutputrun.InputRun* property), 158  
 (*fastr.execution.inputoutputrun.SubInputRun* property), 165  
 (*fastr.planning.inputoutput.Input* property), 227  
 (*fastr.planning.inputoutput.MacroInput* property), 228  
 (*fastr.planning.inputoutput.SubInput* property), 234  
 (*fastr.execution.noderun.NodeRun* property), 189  
 (*fastr.planning.inputgroupcombiner.BaseInputGroupCombiner* property), 218  
 (*fastr.planning.node.Node* property), 250  
InputDict (class in *fastr.planning.node*), 247  
InputGroup (class in *fastr.planning.inputgroup*), 216  
InputRun (class in *fastr.execution.inputoutputrun*), 156  
inputs (*fastr.api.Node* property), 82  
inputs (*fastr.core.interface.Interface* property), 107  
inputs (*fastr.core.tool.Tool* property), 130  
inputs (*fastr.planning.node.Node* attribute), 250  
inputs (*fastr.plugins.FastrInterface* property), 263  
inputs (*fastr.plugins.FlowInterface* property), 266  
inputs (*fastr.plugins.NipypeInterface* property), 272  
InputSpec (class in *fastr.core.interface*), 106  
InputSpecBase (in module *fastr.core.interface*), 106  
insert() (*fastr.execution.inputoutputrun.InputRun* method), 158  
insert() (*fastr.planning.inputoutput.Input* method), 227  
IntCardinalitySpec (class in *fastr.core.cardinality*), 100  
Interface (class in *fastr.core.interface*), 106  
InterfacePluginManager (class in *fastr.plugins.managers.interfacemanager*), 291  
InterfaceResult (class in *fastr.core.interface*), 108  
IOPlugin (class in *fastr.core.ioplugin*), 108  
IOPluginManager (class in *fastr.plugins.managers.iopluginmanager*), 292  
is\_mapping (*fastr.core.samples.SampleValue* property), 125  
is\_sequence (*fastr.core.samples.SampleValue* property), 125  
is\_valid() (*fastr.planning.network.Network* method), 244  
isdatatype() (*fastr.datatypes.DataTypeManager* static method), 145  
isinstance() (*fastr.datatypes.BaseDataType* class method), 142  
isinstance() (*fastr.datatypes.TypeGroup* class method), 149  
isloaded() (*fastr.execution.environmentmodules.EnvironmentModules* method), 153  
isslice (*fastr.core.samples.SampleIndex* property), 120  
isurl() (*fastr.core.ioplugin.IOPlugin* static method), 109  
isurl() (in module *fastr.data.url*), 138  
item\_index (*fastr.execution.inputoutputrun.NamedSubInputRun* property), 160  
item\_index (*fastr.execution.inputoutputrun.SubInputRun* property), 166  
item\_index (*fastr.planning.inputoutput.BaseInput* property), 222  
item\_index (*fastr.planning.inputoutput.NamedSubInput* property), 229  
item\_index (*fastr.planning.inputoutput.SubInput* property), 234  
items() (*fastr.core.samples.HasSamples* method), 115  
items\_prevalidate() (in module *fastr.helpers.jsonschemaresolver*), 209  
iter\_input\_groups() (*fastr.planning.inputgroupcombiner.BaseInputGroupCombiner* method), 218  
iter\_input\_groups() (*fastr.planning.inputgroupcombiner.DefaultInputGroupCombiner* method), 219  
iter\_input\_groups() (*fastr.planning.inputgroupcombiner.MergingInputGroupCombiner* method), 220  
iterconvergingindices() (*fastr.execution.inputoutputrun.OutputRun* method), 162  
iterelements() (*fastr.core.samples.SampleValue* method), 125  
iterinputvalues (*fastr.planning.inputgroup.InputGroup* property), 217  
iteritems() (*fastr.core.samples.HasSamples* method), 115  
iteritems() (*fastr.execution.inputoutputrun.SubInputRun* method), 166  
iteritems() (*fastr.planning.inputoutput.SubInput* method), 234  
itersubinputs() (*fastr.execution.inputoutputrun.BaseInputRun* method), 156  
itersubinputs() (*fastr.execution.inputoutputrun.InputRun* method), 158  
itersubinputs() (*fastr.execution.inputoutputrun.SubInputRun* method), 166  
itersubinputs() (*fastr.planning.inputoutput.BaseInput* method), 222  
itersubinputs() (*fastr.planning.inputoutput.Input* method), 227  
itersubinputs() (*fastr.planning.inputoutput.SubInput* method), 234



## J

[Job](#) (class in `fastr.execution.job`), 169  
[job](#) (`fastr.datatypes.Deferred` property), 147  
[job\\_cleanup\\_level](#) (`fastr.helpers.configmanager.Config` property), 203  
[job\\_finished\(\)](#) (`fastr.execution.networkrun.NetworkRun` method), 187  
[job\\_finished\(\)](#) (`fastr.helpers.filesynchelper.FileSyncHelper` method), 207  
[job\\_finished\(\)](#) (`fastr.plugins.executionplugin.ExecutionPlugin` method), 289  
[job\\_finished\\_callback\(\)](#) (`fastr.plugins.ProcessPoolExecution` method), 274  
[job\\_status\\_check\(\)](#) (`fastr.plugins.SlurmExecution` method), 279  
[job\\_updated](#) (`fastr.helpers.events.EventType` attribute), 206  
[job\\_updated\(\)](#) (`fastr.plugins.ElasticsearchReporter` method), 259  
[job\\_updated\(\)](#) (`fastr.plugins.PimReporter` method), 273  
[job\\_updated\(\)](#) (`fastr.plugins.reportingplugin.ReportingPlugin` method), 290  
[JobAction](#) (class in `fastr.plugins.executionplugin`), 289  
[JobCleanupLevel](#) (class in `fastr.execution.job`), 174  
[jobs](#) (`fastr.core.samples.SampleItemBase` property), 122  
[JobState](#) (class in `fastr.execution.job`), 174  
[join\(\)](#) (in module `fastr.data.url`), 138  
[json](#) (in module `fastr.plugins`), 286

## L

[LazyModule](#) (class in `fastr.helpers.lazy_module`), 210  
[LinearExecution](#) (class in `fastr.plugins`), 267  
[linearized](#) (`fastr.execution.inputoutputrun.SourceOutputRun` property), 164  
[linearized](#) (`fastr.planning.inputoutput.SourceOutput` property), 232  
[Link](#) (class in `fastr.api`), 81  
[Link](#) (class in `fastr.planning.link`), 237  
[link\\_or\\_copy\(\)](#) (in module `fastr.helpers.iohelpers`), 208  
[LinkRun](#) (class in `fastr.execution.linkrun`), 179  
[listeners](#) (`fastr.execution.inputoutputrun.OutputRun` property), 162  
[listeners](#) (`fastr.execution.inputoutputrun.SubOutputRun` property), 168  
[listeners](#) (`fastr.execution.noderun.NodeRun` property), 189  
[listeners](#) (`fastr.planning.inputoutput.Output` property), 231  
[listeners](#) (`fastr.planning.inputoutput.SubOutput` property), 236  
[listeners](#) (`fastr.planning.node.Node` property), 250  
[lmod](#) (`fastr.execution.environmentmodules.ModuleSystem` attribute), 154  
[load\(\)](#) (`fastr.api.Network` class method), 81  
[load\(\)](#) (`fastr.execution.environmentmodules.EnvironmentModules` method), 153  
[load\(\)](#) (`fastr.helpers.filesynchelper.FileSyncHelper` method), 207  
[load\(\)](#) (in module `fastr.helpers.xmltodict`), 216  
[load\\_gpickle\(\)](#) (in module `fastr.helpers.iohelpers`), 208  
[load\\_json\(\)](#) (in module `fastr.helpers.iohelpers`), 208  
[loaded\\_modules](#) (`fastr.execution.environmentmodules.EnvironmentModules` property), 153  
[loads\(\)](#) (in module `fastr.helpers.xmltodict`), 216  
[LocalBinaryTarget](#) (class in `fastr.plugins`), 268  
[lock\\_dir](#) (`fastr.helpers.lockfile.DirectoryLock` property), 211  
[lock\\_dir\\_name](#) (`fastr.helpers.lockfile.DirectoryLock` attribute), 211  
[log\\_record\\_emitted](#) (`fastr.helpers.events.EventType` attribute), 206  
[log\\_record\\_emitted\(\)](#) (`fastr.plugins.PimReporter` method), 273  
[log\\_record\\_emitted\(\)](#) (`fastr.plugins.reportingplugin.ReportingPlugin` method), 290  
[log\\_to\\_file](#) (`fastr.helpers.configmanager.Config` property), 203  
[logdir](#) (`fastr.helpers.configmanager.Config` property), 203  
[logfile](#) (`fastr.execution.job.Job` property), 172  
[logging\\_config](#) (`fastr.helpers.configmanager.Config` property), 203  
[loglevel](#) (`fastr.helpers.configmanager.Config` property), 203  
[logtype](#) (`fastr.helpers.configmanager.Config` property), 203  
[logurl](#) (`fastr.execution.job.Job` property), 172  
[long\\_id](#) (`fastr.execution.networkrun.NetworkRun` property), 187  
[lookup\(\)](#) (`fastr.datatypes.Deferred` class method), 147

## M

[MacroInput](#) (class in `fastr.planning.inputoutput`), 228  
[MacroNode](#) (class in `fastr.planning.node`), 247  
[MacroNodeRun](#) (class in `fastr.execution.macronoderun`), 182  
[MacroOutput](#) (class in `fastr.planning.inputoutput`), 228  
[MacroOutputRun](#) (class in `fastr.execution.inputoutputrun`), 159  
[MacroTarget](#) (class in `fastr.plugins`), 269  
[main\(\)](#) (in module `fastr.execution.executionscript`), 154  
[main\(\)](#) (in module `fastr.helpers.clear_pycs`), 199  
[main\(\)](#) (in module `fastr.utils.cmd`), 305

[main\(\)](#) (in module `fastr.utils.cmd.cat`), 305  
[main\(\)](#) (in module `fastr.utils.cmd.dump`), 305  
[main\(\)](#) (in module `fastr.utils.cmd.execute`), 305  
[main\(\)](#) (in module `fastr.utils.cmd.extract_argparse`), 306  
[main\(\)](#) (in module `fastr.utils.cmd.provenance`), 306  
[main\(\)](#) (in module `fastr.utils.cmd.pylint`), 306  
[main\(\)](#) (in module `fastr.utils.cmd.report`), 306  
[main\(\)](#) (in module `fastr.utils.cmd.run`), 306  
[main\(\)](#) (in module `fastr.utils.cmd.sink`), 307  
[main\(\)](#) (in module `fastr.utils.cmd.source`), 307  
[main\(\)](#) (in module `fastr.utils.cmd.test`), 307  
[main\(\)](#) (in module `fastr.utils.cmd.trace`), 307  
[main\(\)](#) (in module `fastr.utils.cmd.upgrade`), 308  
[main\(\)](#) (in module `fastr.utils.cmd.verify`), 309  
[main\(\)](#) (in module `fastr.utils.gettools`), 303  
[major](#) (`fastr.core.version.Version` property), 133  
[make\\_file\\_promise\(\)](#) (`fastr.helpers.filesynchelper.FileSyncHelper` method), 207  
[mapping\\_part\(\)](#) (`fastr.core.samples.SampleValue` method), 125  
[match\\_types\(\)](#) (`fastr.datatypes.DataTypeManager` method), 146  
[match\\_types\\_any\(\)](#) (`fastr.datatypes.DataTypeManager` method), 146  
[MaxCardinalitySpec](#) (class in `fastr.core.cardinality`), 101  
[md5\\_checksum\(\)](#) (in module `fastr.helpers.checksum`), 197  
[members](#) (`fastr.datatypes.TypeGroup` attribute), 149  
[memory](#) (`fastr.api.ResourceLimit` property), 96  
[memory](#) (`fastr.core.resourcelimit.ResourceLimit` property), 113  
[merge\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[merge\(\)](#) (`fastr.planning.inputgroupcombiner.DefaultInputGroupCombiner` method), 219  
[merge\(\)](#) (`fastr.planning.inputgroupcombiner.MergingInputGroupCombiner` method), 220  
[merge\\_default\(\)](#) (`fastr.helpers.configmanager.EmptyDefault` method), 205  
[merge\\_dimensions](#) (`fastr.execution.noderun.NodeRun` property), 189  
[merge\\_dimensions](#) (`fastr.planning.node.Node` property), 250  
[merge\\_failed\\_annotations\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[merge\\_payloads\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[merge\\_sample\\_data\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[merge\\_sample\\_id\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[merge\\_sample\\_index\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[merge\\_sample\\_jobs\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[merge\\_sample\\_status\(\)](#) (`fastr.planning.inputgroupcombiner.BaseInputGroupCombiner` method), 218  
[MergingInputGroupCombiner](#) (class in `fastr.planning.inputgroupcombiner`), 219  
[MinCardinalitySpec](#) (class in `fastr.core.cardinality`), 101  
[minor](#) (`fastr.core.version.Version` property), 133  
[Missing](#) (class in `fastr.datatypes`), 148  
[MISSING](#) (`fastr.core.samples.SampleState` attribute), 123  
[module](#)  
[fastr.\\_\\_init\\_\\_](#), 85  
[fastr.api](#), 95  
[fastr.core](#), 97  
[fastr.core.cardinality](#), 97  
[fastr.core.dimension](#), 104  
[fastr.core.interface](#), 106  
[fastr.core.ioplugin](#), 108  
[fastr.core.provenance](#), 111  
[fastr.core.resourcelimit](#), 112  
[fastr.core.samples](#), 114  
[fastr.core.target](#), 125  
[fastr.core.test](#), 136  
[fastr.core.tool](#), 128  
[fastr.core.version](#), 131  
[fastr.core.vfs](#), 133  
[fastr.data](#), 137  
[fastr.data.url](#), 137  
[fastr.datatypes](#), 139  
[fastr.exceptions](#), 86  
[fastr.execution](#), 151  
[fastr.execution.basenoderun](#), 151  
[fastr.execution.environmentmodules](#), 152  
[fastr.execution.executionscript](#), 154  
[fastr.execution.flownoderun](#), 154  
[fastr.execution.inputoutputrun](#), 155  
[fastr.execution.job](#), 168  
[fastr.execution.linkrun](#), 179  
[fastr.execution.macronoderun](#), 182  
[fastr.execution.networkanalyzer](#), 183  
[fastr.execution.networkchunker](#), 184  
[fastr.execution.networkrun](#), 185  
[fastr.execution.noderun](#), 187  
[fastr.execution.sinknoderun](#), 190  
[fastr.execution.sourcenoderun](#), 192  
[fastr.globals](#), 94  
[fastr.helpers](#), 195

- `fastr.helpers.checksum`, 197
  - `fastr.helpers.classproperty`, 198
  - `fastr.helpers.clear_pycs`, 199
  - `fastr.helpers.configmanager`, 199
  - `fastr.helpers.events`, 206
  - `fastr.helpers.filesynchelper`, 207
  - `fastr.helpers.iohelpers`, 208
  - `fastr.helpers.jsonschemaparser`, 208
  - `fastr.helpers.lazy_module`, 210
  - `fastr.helpers.lockfile`, 211
  - `fastr.helpers.procutils`, 212
  - `fastr.helpers.report`, 212
  - `fastr.helpers.rest_generation`, 212
  - `fastr.helpers.schematotable`, 212
  - `fastr.helpers.shellescape`, 213
  - `fastr.helpers.sysinfo`, 214
  - `fastr.helpers.xmltodict`, 215
  - `fastr.planning`, 216
  - `fastr.planning.inputgroup`, 216
  - `fastr.planning.inputgroupcombiner`, 218
  - `fastr.planning.inputoutput`, 220
  - `fastr.planning.link`, 237
  - `fastr.planning.network`, 240
  - `fastr.planning.node`, 244
  - `fastr.plugins`, 253
  - `fastr.plugins.executionplugin`, 287
  - `fastr.plugins.managers`, 290
  - `fastr.plugins.managers.executionpluginmanager`, 290
  - `fastr.plugins.managers.interfacemanager`, 291
  - `fastr.plugins.managers.iopluginmanager`, 292
  - `fastr.plugins.managers.networkmanager`, 294
  - `fastr.plugins.managers.objectmanager`, 295
  - `fastr.plugins.managers.pluginmanager`, 297
  - `fastr.plugins.managers.targetmanager`, 300
  - `fastr.plugins.managers.toolmanager`, 300
  - `fastr.plugins.reportingplugin`, 290
  - `fastr.test`, 301
  - `fastr.utils`, 301
  - `fastr.utils.cmd`, 305
  - `fastr.utils.cmd.cat`, 305
  - `fastr.utils.cmd.dump`, 305
  - `fastr.utils.cmd.execute`, 305
  - `fastr.utils.cmd.extract_argparse`, 305
  - `fastr.utils.cmd.provenance`, 306
  - `fastr.utils.cmd.pylint`, 306
  - `fastr.utils.cmd.report`, 306
  - `fastr.utils.cmd.run`, 306
  - `fastr.utils.cmd.sink`, 307
  - `fastr.utils.cmd.source`, 307
  - `fastr.utils.cmd.test`, 307
  - `fastr.utils.cmd.trace`, 307
  - `fastr.utils.cmd.upgrade`, 308
  - `fastr.utils.cmd.verify`, 309
  - `fastr.utils.compare`, 302
  - `fastr.utils.dicteq`, 302
  - `fastr.utils.gettools`, 303
  - `fastr.utils.multiprocesswrapper`, 303
  - `fastr.utils.verify`, 303
  - `fastr.version`, 94
  - module (`fastr.plugins.BlockingExecution` attribute), 254
  - module (`fastr.plugins.CommaSeperatedValueFile` attribute), 255
  - module (`fastr.plugins.CrossValidation` attribute), 256
  - module (`fastr.plugins.DockerTarget` attribute), 259
  - module (`fastr.plugins.DRMAAExecution` attribute), 258
  - module (`fastr.plugins.ElasticsearchReporter` attribute), 259
  - module (`fastr.plugins.FastrInterface` attribute), 263
  - module (`fastr.plugins.FileSystem` attribute), 264
  - module (`fastr.plugins.FlowInterface` attribute), 266
  - module (`fastr.plugins.HTTPPlugin` attribute), 267
  - module (`fastr.plugins.LinearExecution` attribute), 267
  - module (`fastr.plugins.LocalBinaryTarget` attribute), 269
  - module (`fastr.plugins.MacroTarget` attribute), 270
  - module (`fastr.plugins.NetworkScope` attribute), 270
  - module (`fastr.plugins.NipypeInterface` attribute), 272
  - module (`fastr.plugins.Null` attribute), 272
  - module (`fastr.plugins.PimReporter` attribute), 273
  - module (`fastr.plugins.ProcessPoolExecution` attribute), 274
  - module (`fastr.plugins.Reference` attribute), 276
  - module (`fastr.plugins.RQExecution` attribute), 275
  - module (`fastr.plugins.S3Filesystem` attribute), 277
  - module (`fastr.plugins.SimpleReport` attribute), 278
  - module (`fastr.plugins.SingularityTarget` attribute), 278
  - module (`fastr.plugins.SlurmExecution` attribute), 279
  - module (`fastr.plugins.StrongrExecution` attribute), 280
  - module (`fastr.plugins.VirtualFileSystem` attribute), 281
  - module (`fastr.plugins.VirtualFileSystemRegularExpression` attribute), 282
  - module (`fastr.plugins.VirtualFileSystemValueList` attribute), 283
  - module (`fastr.plugins.XNATStorage` attribute), 286
  - `ModuleSystem` (class in `fastr.execution.environmentmodules`), 154
  - `monitor_docker()` (`fastr.plugins.DockerTarget` method), 259
  - `monitor_process()` (`fastr.core.target.SubprocessBasedTarget` method), 126
  - `mounts` (`fastr.helpers.configmanager.Config` property), 203
- N**
- `n_current_jobs` (`fastr.plugins.DRMAAExecution` prop-

- [erty](#)), 258
  - [name \(fastr.core.dimension.Dimension property\)](#), 104
  - [name \(fastr.core.tool.Tool attribute\)](#), 130
  - [name \(fastr.datatypes.BaseDataType attribute\)](#), 142
  - [name \(fastr.execution.noderun.NodeRun property\)](#), 190
  - [name \(fastr.planning.node.Node property\)](#), 250
  - [NamedSubInput \(class in fastr.planning.inputoutput\)](#), 228
  - [NamedSubinputRun \(class in fastr.execution.inputoutputrun\)](#), 159
  - [namedtuple\\_to\\_dict\(\) \(in module fastr.helpers.sysinfo\)](#), 215
  - [namespace \(fastr.core.tool.Tool attribute\)](#), 130
  - [namespace \(fastr.planning.network.Network attribute\)](#), 244
  - [NATIVE\\_SPEC \(fastr.plugins.DRMAAExecution attribute\)](#), 256
  - [ndims \(fastr.core.dimension.HasDimensions property\)](#), 106
  - [ndims \(fastr.core.samples.SampleCollection property\)](#), 118
  - [ndims \(fastr.execution.inputoutputrun.SourceOutputRun property\)](#), 164
  - [Network \(class in fastr.api\)](#), 78
  - [Network \(class in fastr.planning.network\)](#), 240
  - [network \(fastr.execution.networkrun.NetworkRun property\)](#), 187
  - [network \(fastr.planning.node.MacroNode property\)](#), 248
  - [NETWORK\\_DUMP\\_FILE\\_NAME \(fastr.execution.networkrun.NetworkRun attribute\)](#), 185
  - [NETWORK\\_DUMP\\_FILE\\_NAME \(fastr.planning.network.Network attribute\)](#), 240
  - [network\\_run \(fastr.execution.macronoderun.MacroNodeRuns property\)](#), 183
  - [NetworkAnalyzer \(class in fastr.execution.networkanalyzer\)](#), 183
  - [NetworkChunker \(class in fastr.execution.networkchunker\)](#), 184
  - [NetworkManager \(class in fastr.plugins.managers.networkmanager\)](#), 294
  - [NetworkRun \(class in fastr.execution.networkrun\)](#), 185
  - [networks \(fastr attribute\)](#), 78
  - [networks\\_path \(fastr.helpers.configmanager.Config property\)](#), 203
  - [NetworkScope \(class in fastr.plugins\)](#), 270
  - [NipypeInterface \(class in fastr.plugins\)](#), 271
  - [no\\_cleanup \(fastr.execution.job.JobCleanupLevel attribute\)](#), 174
  - [Node \(class in fastr.api\)](#), 82
  - [Node \(class in fastr.planning.node\)](#), 248
  - [node \(fastr.core.cardinality.AsCardinalitySpec property\)](#), 98
  - [node \(fastr.core.cardinality.ValueCardinalitySpec property\)](#), 103
  - [node \(fastr.execution.inputoutputrun.SubInputRun property\)](#), 166
  - [node \(fastr.execution.inputoutputrun.SubOutputRun property\)](#), 168
  - [node \(fastr.planning.inputoutput.BaseInputOutput property\)](#), 224
  - [node \(fastr.planning.inputoutput.SubInput property\)](#), 234
  - [node \(fastr.planning.inputoutput.SubOutput property\)](#), 236
  - [node\\_class \(fastr.core.tool.Tool attribute\)](#), 130
  - [NODE\\_RUN\\_MAP \(fastr.execution.basenoderun.BaseNodeRun attribute\)](#), 151
  - [NODE\\_RUN\\_TYPES \(fastr.execution.basenoderun.BaseNodeRun attribute\)](#), 151
  - [NODE\\_TYPES \(fastr.planning.node.BaseNode attribute\)](#), 245
  - [nodegroup \(fastr.planning.node.Node property\)](#), 250
  - [nodegroup \(fastr.planning.node.SourceNode property\)](#), 253
  - [nodegroups \(fastr.execution.networkrun.NetworkRun property\)](#), 187
  - [nodegroups \(fastr.planning.network.Network property\)](#), 244
  - [NodeRun \(class in fastr.execution.noderun\)](#), 187
  - [non\\_failed \(fastr.execution.job.JobCleanupLevel attribute\)](#), 174
  - [nonexistent \(fastr.execution.job.JobState attribute\)](#), 176
  - [normurl\(\) \(in module fastr.data.url\)](#), 139
  - [not\\_draft4\(\) \(in module fastr.helpers.jsonschemaresolver\)](#), 209
  - [ns\\_id \(fastr.core.tool.Tool property\)](#), 130
  - [ns\\_id \(fastr.planning.network.Network property\)](#), 244
  - [Null \(class in fastr.plugins\)](#), 272
- ## O
- [object\\_class \(fastr.plugins.managers.networkmanager.NetworkManager property\)](#), 295
  - [object\\_class \(fastr.plugins.managers.objectmanager.ObjectManager property\)](#), 296
  - [object\\_class \(fastr.plugins.managers.toolmanager.ToolManager property\)](#), 301
  - [ObjectManager \(class in fastr.plugins.managers.objectmanager\)](#), 295
  - [objectversions\(\) \(fastr.plugins.managers.objectmanager.ObjectManager method\)](#), 296
  - [one\\_of\\_draft4\(\) \(in module fastr.helpers.jsonschemaresolver\)](#), 209
  - [options \(fastr.datatypes.EnumType attribute\)](#), 148
  - [Output \(class in fastr.api\)](#), 84
  - [Output \(class in fastr.planning.inputoutput\)](#), 229



output (*fastr.api.Node* property), 82  
 output (*fastr.execution.sourcenoderun.SourceNodeRun* property), 194  
 output (*fastr.planning.node.SourceNode* property), 253  
 OutputDict (class in *fastr.planning.node*), 251  
 OutputRun (class in *fastr.execution.inputoutputrun*), 160  
 outputs (*fastr.api.Node* property), 82  
 outputs (*fastr.core.interface.Interface* property), 107  
 outputs (*fastr.core.tool.Tool* property), 130  
 outputs (*fastr.planning.node.Node* attribute), 250  
 outputs (*fastr.plugins.FastrInterface* property), 263  
 outputs (*fastr.plugins.FlowInterface* property), 266  
 outputs (*fastr.plugins.NipypeInterface* property), 272  
 outputsize (*fastr.execution.flownoderun.FlowNodeRun* property), 155  
 outputsize (*fastr.execution.noderun.NodeRun* property), 190  
 outputsize (*fastr.execution.sourcenoderun.SourceNodeRun* property), 194  
 outputsize (*fastr.planning.node.FlowNode* property), 246  
 outputsize (*fastr.planning.node.Node* property), 250  
 OutputSpec (class in *fastr.core.interface*), 108  
 OutputSpecBase (in module *fastr.core.interface*), 108

## P

parent (*fastr.core.samples.SampleCollection* property), 118  
 parent (*fastr.datatypes.BaseDataType* attribute), 142  
 parent (*fastr.execution.linkrun.LinkRun* property), 182  
 parent (*fastr.execution.noderun.NodeRun* property), 190  
 parent (*fastr.planning.inputgroup.InputGroup* property), 217  
 parent (*fastr.planning.link.Link* property), 239  
 parent (*fastr.planning.node.Node* property), 250  
 parse() (*fastr.helpers.schematotable.SchemaPrinter* method), 213  
 parse\_uri() (*fastr.plugins.XNATStorage* method), 286  
 parsed\_value (*fastr.datatypes.BaseDataType* property), 142  
 parsed\_value (*fastr.datatypes.Deferred* property), 147  
 parsed\_value (*fastr.datatypes.URLType* property), 150  
 path (*fastr.core.tool.Tool* property), 130  
 path (in module *fastr.plugins*), 286  
 path\_to\_url() (*fastr.core.ioplugin.IOPlugin* method), 109  
 path\_to\_url() (*fastr.core.vfs.VirtualFileSystem* method), 135  
 path\_to\_url() (*fastr.plugins.FileSystem* method), 264  
 paths (*fastr.plugins.LocalBinaryTarget* property), 269  
 pattern\_properties\_prevalid() (in module *fastr.helpers.jsonschemasparser*), 210  
 pid\_file (*fastr.helpers.lockfile.DirectoryLock* property), 212

pid\_file\_name (*fastr.helpers.lockfile.DirectoryLock* attribute), 212  
 pim\_batch\_size (*fastr.helpers.configmanager.Config* property), 203  
 pim\_debug (*fastr.helpers.configmanager.Config* property), 203  
 pim\_finished\_timeout (*fastr.helpers.configmanager.Config* property), 203  
 pim\_host (*fastr.helpers.configmanager.Config* property), 203  
 pim\_update\_interval (*fastr.helpers.configmanager.Config* property), 203  
 pim\_username (*fastr.helpers.configmanager.Config* property), 203  
 PimReporter (class in *fastr.plugins*), 273  
 plugin\_class (*fastr.datatypes.DataTypeManager* property), 146  
 plugin\_class (*fastr.plugins.managers.pluginmanager.PluginManager* property), 297  
 plugin\_class (*fastr.plugins.managers.pluginmanager.PluginSubManager* property), 298  
 PluginManager (class in *fastr.plugins.managers.pluginmanager*), 297  
 plugins\_path (*fastr.helpers.configmanager.Config* property), 203  
 PluginSubManager (class in *fastr.plugins.managers.pluginmanager*), 298  
 PluginsView (class in *fastr.plugins.managers.pluginmanager*), 298  
 poll\_datatype() (*fastr.datatypes.DataTypeManager* method), 146  
 populate() (*fastr.datatypes.DataTypeManager* method), 146  
 populate() (*fastr.plugins.managers.toolmanager.ToolManager* method), 301  
 predefined (*fastr.core.cardinality.AsCardinalitySpec* property), 98  
 predefined (*fastr.core.cardinality.CardinalitySpec* property), 100  
 predefined (*fastr.core.cardinality.IntCardinalitySpec* property), 101  
 preference (*fastr.datatypes.TypeGroup* attribute), 149  
 preferred\_types (*fastr.datatypes.DataTypeManager* property), 146  
 preferred\_types (*fastr.execution.inputoutputrun.OutputRun* property), 162  
 preferred\_types (*fastr.execution.inputoutputrun.SubOutputRun* property), 168  
 preferred\_types (*fastr.helpers.configmanager.Config* property), 203  
 preferred\_types (*fastr.planning.inputoutput.Output* property), 231

- `preferred_types` (*fastr.planning.inputoutput.SubOutput* property), 236
- `prepend()` (*fastr.helpers.configmanager.EmptyDefault* method), 205
- `primary` (*fastr.planning.inputgroup.InputGroup* property), 217
- `print_help()` (in module *fastr.utils.cmd*), 305
- `print_job_result()` (in module *fastr.helpers.report*), 212
- `print_result()` (*fastr.core.ioplugin.IOPlugin* static method), 110
- `print_sample_sink()` (in module *fastr.utils.cmd.trace*), 307
- `print_samples()` (in module *fastr.utils.cmd.trace*), 307
- `print_sinks()` (in module *fastr.utils.cmd.trace*), 307
- `print_value` (*fastr.planning.node.ConstantNode* property), 246
- `printlines()` (*fastr.helpers.schematatable.SchemaPrinter* method), 213
- `process_callbacks()` (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289
- `process_pool_worker_number` (*fastr.helpers.configmanager.Config* property), 203
- `processing_callback` (*fastr.execution.job.JobState* attribute), 176
- `ProcessPoolExecution` (class in *fastr.plugins*), 274
- `ProcessUsageCollection` (class in *fastr.core.target*), 125
- `properties_postvalidate()` (in module *fastr.helpers.jsonschemaresolver*), 210
- `properties_prevalidate()` (in module *fastr.helpers.jsonschemaresolver*), 210
- `protected_modules` (*fastr.helpers.configmanager.Config* property), 203
- `PROV_DUMP` (*fastr.execution.job.Job* attribute), 169
- `Provenance` (class in *fastr.core.provenance*), 111
- `provenance` (*fastr.datatypes.Deferred* property), 147
- `provfile` (*fastr.execution.job.Job* property), 172
- `provurl` (*fastr.execution.job.Job* property), 172
- `pull_source_data()` (*fastr.core.ioplugin.IOPlugin* method), 110
- `pull_source_data()` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), 293
- `push_sink_data()` (*fastr.core.ioplugin.IOPlugin* method), 110
- `push_sink_data()` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), 293
- `push_sink_data()` (*fastr.plugins.Reference* method), 276
- `put_url()` (*fastr.core.ioplugin.IOPlugin* method), 110
- `put_url()` (*fastr.core.vfs.VirtualFileSystem* method), 136
- `put_url()` (*fastr.plugins.FileSystem* method), 264
- `put_url()` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), 293
- `put_url()` (*fastr.plugins.Null* method), 272
- `put_url()` (*fastr.plugins.S3Filesystem* method), 277
- `put_url()` (*fastr.plugins.XNATStorage* method), 286
- `put_value()` (*fastr.core.ioplugin.IOPlugin* method), 110
- `put_value()` (*fastr.core.vfs.VirtualFileSystem* method), 136
- `put_value()` (*fastr.plugins.FileSystem* method), 264
- `put_value()` (*fastr.plugins.Null* method), 273
- `put_value()` (*fastr.plugins.S3Filesystem* method), 277
- Python Enhancement Proposals
- PEP 8, 28
  - PEP 8#class-names, 28
  - PEP 8#global-variable-names, 28
  - PEP 8#method-names-and-instance-variables, 28
  - PEP 8#package-and-module-names, 28
  - PEP 8#prescriptive-naming-conventions, 28
- ## Q
- `queue` (*fastr.plugins.executionplugin.JobAction* attribute), 290
- `queue_job()` (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289
- `queue_report_interval` (*fastr.helpers.configmanager.Config* property), 203
- `queued` (*fastr.execution.job.JobState* attribute), 176
- `quote_argument()` (in module *fastr.helpers.shellescape*), 213
- ## R
- `RangeCardinalitySpec` (class in *fastr.core.cardinality*), 102
- `raw_value` (*fastr.datatypes.BaseDataType* property), 142
- `read_bytes` (*fastr.core.target.SystemUsageInfo* property), 127
- `read_config()` (*fastr.helpers.configmanager.Config* method), 204
- `read_config_files` (*fastr.helpers.configmanager.Config* attribute), 204
- `read_config_string()` (*fastr.helpers.configmanager.Config* method), 204
- `read_sink_data()` (in module *fastr.utils.cmd.trace*), 307
- `readfastrschema()` (*fastr.helpers.jsonschemaresolver.FastrRefResolver* static method), 208
- `readfile()` (*fastr.helpers.jsonschemaresolver.FastrRefResolver* static method), 208
- `Reference` (class in *fastr.plugins*), 275
- `references` (*fastr.core.tool.Tool* attribute), 130

- [register\\_fields\(\)](#) (*fastr.helpers.configmanager.Config method*), 204  
[register\\_job\(\)](#) (*fastr.plugins.executionplugin.ExecutionPlugin method*), 289  
[register\\_listener\(\)](#) (in module *fastr.helpers.events*), 206  
[register\\_signals\(\)](#) (*fastr.execution.networkrun.NetworkRun method*), 187  
[register\\_url\\_scheme\(\)](#) (*fastr.plugins.managers.iopluginmanager.IOPuginManager static method*), 294  
[register\\_url\\_scheme\(\)](#) (in module *fastr.data.url*), 139  
[regression\\_check\(\)](#) (*fastr.plugins.DRMAAExecution method*), 258  
[release\(\)](#) (*fastr.helpers.lockfile.DirectoryLock method*), 212  
[release\\_job\(\)](#) (*fastr.plugins.executionplugin.ExecutionPlugin method*), 289  
[reload\(\)](#) (*fastr.execution.environmentmodules.EnvironmentModules method*), 153  
[remove\(\)](#) (*fastr.execution.inputoutputrun.InputRun method*), 159  
[remove\(\)](#) (*fastr.planning.inputoutput.Input method*), 228  
[remove\(\)](#) (*fastr.planning.inputoutput.SubInput method*), 234  
[remove\(\)](#) (*fastr.planning.network.Network method*), 244  
[remove\\_listener\(\)](#) (in module *fastr.helpers.events*), 206  
[replace\(\)](#) (*fastr.core.samples.SampleItemBase method*), 122  
[reporting\\_plugins](#) (*fastr.helpers.configmanager.Config property*), 204  
[ReportingPlugin](#) (class in *fastr.plugins.reportingplugin*), 290  
[required](#) (*fastr.planning.inputoutput.BaseInputOutput property*), 224  
[requirements](#) (*fastr.core.tool.Tool attribute*), 130  
[ResourceLimit](#) (class in *fastr.api*), 95  
[ResourceLimit](#) (class in *fastr.core.resourcelimit*), 112  
[resources](#) (*fastr.execution.job.Job property*), 172  
[resources](#) (*fastr.execution.noderun.NodeRun property*), 190  
[resourcesdir](#) (*fastr.helpers.configmanager.Config property*), 204  
[RESULT\\_DUMP](#) (*fastr.execution.job.Job attribute*), 169  
[resulting\\_datatype](#) (*fastr.execution.inputoutputrun.OutputRun property*), 162  
[resulting\\_datatype](#) (*fastr.execution.inputoutputrun.SubOutputRun property*), 168  
[resulting\\_datatype](#) (*fastr.planning.inputoutput.Output property*), 231  
[resulting\\_datatype](#) (*fastr.planning.inputoutput.SubOutput property*), 236  
[rmem](#) (*fastr.core.target.SystemUsageInfo property*), 127  
[RQExecution](#) (class in *fastr.plugins*), 275  
[run\\_command\(\)](#) (*fastr.core.target.Target method*), 127  
[run\\_command\(\)](#) (*fastr.plugins.DockerTarget method*), 259  
[run\\_command\(\)](#) (*fastr.plugins.LocalBinaryTarget method*), 269  
[run\\_command\(\)](#) (*fastr.plugins.MacroTarget method*), 270  
[run\\_command\(\)](#) (*fastr.plugins.SingularityTarget method*), 278  
[run\\_finished](#) (*fastr.helpers.events.EventType attribute*), 206  
[run\\_finished\(\)](#) (*fastr.plugins.PimReporter method*), 274  
[run\\_finished\(\)](#) (*fastr.plugins.reportingplugin.ReportingPlugin method*), 290  
[run\\_finished\(\)](#) (*fastr.plugins.SimpleReport method*), 278  
[run\\_job\(\)](#) (*fastr.plugins.RQExecution class method*), 275  
[run\\_pylint\(\)](#) (in module *fastr.utils.cmd.pylint*), 306  
[run\\_started](#) (*fastr.helpers.events.EventType attribute*), 206  
[run\\_started\(\)](#) (*fastr.plugins.PimReporter method*), 274  
[run\\_started\(\)](#) (*fastr.plugins.reportingplugin.ReportingPlugin method*), 290  
[running](#) (*fastr.execution.job.JobState attribute*), 176
- ## S
- [S3Filesystem](#) (class in *fastr.plugins*), 276  
[SampleBaseId](#) (class in *fastr.core.samples*), 115  
[SampleCollection](#) (class in *fastr.core.samples*), 116  
[SampleId](#) (class in *fastr.core.samples*), 118  
[SampleIndex](#) (class in *fastr.core.samples*), 119  
[SampleItem](#) (class in *fastr.core.samples*), 120  
[SampleItemBase](#) (class in *fastr.core.samples*), 120  
[SamplePayload](#) (class in *fastr.core.samples*), 123  
[samples](#) (*fastr.core.samples.ContainsSamples property*), 114  
[samples](#) (*fastr.execution.inputoutputrun.OutputRun property*), 162  
[samples](#) (*fastr.execution.inputoutputrun.SubOutputRun property*), 168  
[samples](#) (*fastr.planning.inputoutput.SubOutput property*), 236  
[SampleState](#) (class in *fastr.core.samples*), 123  
[SampleValue](#) (class in *fastr.core.samples*), 124  
[save\(\)](#) (*fastr.api.Network method*), 81  
[save\\_gpickle\(\)](#) (in module *fastr.helpers.iohelpers*), 208  
[save\\_json\(\)](#) (in module *fastr.helpers.iohelpers*), 208  
[save\\_version\(\)](#) (in module *fastr.version*), 95

- SBATCH (*fastr.plugins.SlurmExecution* attribute), 278
- SCANCEL (*fastr.plugins.SlurmExecution* attribute), 279
- schemadir (*fastr.helpers.configmanager.Config* property), 204
- SchemaPrinter (class in *fastr.helpers.schematatable*), 212
- scheme (*fastr.core.ioplugin.IOPlugin* property), 110
- scheme (*fastr.core.vfs.VirtualFileSystem* property), 136
- scheme (*fastr.plugins.CommaSeperatedValueFile* attribute), 255
- scheme (*fastr.plugins.FileSystem* attribute), 264
- scheme (*fastr.plugins.HTTPPlugin* attribute), 267
- scheme (*fastr.plugins.NetworkScope* attribute), 271
- scheme (*fastr.plugins.Null* attribute), 273
- scheme (*fastr.plugins.Reference* attribute), 276
- scheme (*fastr.plugins.S3Filesystem* attribute), 277
- scheme (*fastr.plugins.VirtualFileSystem* attribute), 281
- scheme (*fastr.plugins.VirtualFileSystemRegularExpression* attribute), 282
- scheme (*fastr.plugins.VirtualFileSystemValueList* attribute), 283
- scheme (*fastr.plugins.XNATStorage* attribute), 286
- SCONTROL (*fastr.plugins.SlurmExecution* attribute), 279
- send\_job() (*fastr.plugins.DRMAAExecution* method), 258
- sequence\_part() (*fastr.core.samples.SampleValue* method), 125
- serialize() (*fastr.core.provenance.Provenance* method), 112
- serialize() (*fastr.core.tool.Tool* method), 130
- serialize() (*fastr.datatypes.DataType* method), 143
- server (*fastr.plugins.XNATStorage* property), 286
- set\_current\_run() (in module *fastr.globals*), 94
- set\_data() (*fastr.execution.networkrun.NetworkRun* method), 187
- set\_data() (*fastr.execution.sinknoderun.SinkNodeRun* method), 191
- set\_data() (*fastr.execution.sourcenoderun.ConstantNodeRun* method), 193
- set\_data() (*fastr.execution.sourcenoderun.SourceNodeRun* method), 194
- set\_data() (*fastr.planning.node.ConstantNode* method), 246
- set\_data() (*fastr.planning.node.SourceNode* method), 253
- set\_field() (*fastr.helpers.configmanager.Config* method), 204
- set\_memory\_multiplier() (*fastr.api.ResourceLimit* class method), 96
- set\_memory\_multiplier() (*fastr.core.resourcelimit.ResourceLimit* class method), 113
- set\_result() (*fastr.execution.flownoderun.AdvancedFlowNodeRun* method), 154
- set\_result() (*fastr.execution.flownoderun.FlowNodeRun* method), 155
- set\_result() (*fastr.execution.noderun.NodeRun* method), 190
- set\_result() (*fastr.execution.sinknoderun.SinkNodeRun* method), 192
- setup() (*fastr.core.ioplugin.IOPlugin* method), 111
- setup() (*fastr.core.vfs.VirtualFileSystem* method), 136
- sha1\_checksum() (in module *fastr.helpers.checksum*), 197
- show\_jobs() (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289
- signal\_dependent\_jobs() (*fastr.plugins.executionplugin.ExecutionPlugin* method), 289
- SimpleReport (class in *fastr.plugins*), 277
- SINGULARITY\_BIN (*fastr.plugins.SingularityTarget* attribute), 278
- SingularityTarget (class in *fastr.plugins*), 278
- sink() (in module *fastr.utils.cmd.sink*), 307
- SINK\_DUMP\_FILE\_NAME (*fastr.execution.networkrun.NetworkRun* attribute), 185
- SINK\_DUMP\_FILE\_NAME (*fastr.planning.network.Network* attribute), 240
- SinkJob (class in *fastr.execution.job*), 176
- sinklist (*fastr.execution.networkrun.NetworkRun* property), 187
- SinkNode (class in *fastr.planning.node*), 251
- SinkNodeRun (class in *fastr.execution.sinknoderun*), 190
- size (*fastr.core.dimension.Dimension* property), 104
- size (*fastr.core.dimension.HasDimensions* property), 106
- size (*fastr.execution.inputoutputrun.SourceOutputRun* property), 164
- size (*fastr.execution.linkrun.LinkRun* property), 182
- slurm\_job\_check\_interval (*fastr.helpers.configmanager.Config* property), 204
- slurm\_partition (*fastr.helpers.configmanager.Config* property), 204
- SlurmExecution (class in *fastr.plugins*), 278
- solve\_broadcast() (*fastr.planning.inputgroup.InputGroup* class method), 217
- source (*fastr.core.dimension.ForwardsDimensions* property), 105
- source (*fastr.execution.inputoutputrun.InputRun* property), 159
- source (*fastr.execution.inputoutputrun.SubInputRun* property), 166
- source (*fastr.execution.linkrun.LinkRun* property), 182
- source (*fastr.planning.inputoutput.Input* property), 228
- source (*fastr.planning.inputoutput.SubInput* property),



- 234
- source (*fastr.planning.link.Link* property), 239
- source() (in module *fastr.utils.cmd.source*), 307
- SOURCE\_DUMP\_FILE\_NAME  
(*fastr.execution.networkrun.NetworkRun* attribute), 185
- SOURCE\_DUMP\_FILE\_NAME  
(*fastr.planning.network.Network* attribute), 240
- source\_job\_limit (*fastr.helpers.configmanager.Config* property), 204
- source\_output (*fastr.execution.inputoutputrun.SubInputRun* property), 166
- source\_output (*fastr.planning.inputoutput.SubInput* property), 234
- sourcegroup (*fastr.execution.sourcenoderun.SourceNodeRun* property), 195
- sourcegroup (*fastr.planning.node.SourceNode* property), 253
- SourceJob (class in *fastr.execution.job*), 177
- sourcelist (*fastr.execution.networkrun.NetworkRun* property), 187
- SourceNode (class in *fastr.planning.node*), 252
- SourceNodeRun (class in *fastr.execution.sourcenoderun*), 193
- SourceOutput (class in *fastr.planning.inputoutput*), 231
- SourceOutputRun (class in *fastr.execution.inputoutputrun*), 162
- spec\_fields (*fastr.plugins.DRMAAExecution* property), 258
- split() (in module *fastr.data.url*), 139
- SQUEUE (*fastr.plugins.SlurmExecution* attribute), 279
- SQUEUE\_FORMAT (*fastr.plugins.SlurmExecution* attribute), 279
- status (*fastr.core.samples.SampleItemBase* property), 123
- status (*fastr.core.version.Version* property), 133
- status (*fastr.execution.job.Job* property), 172
- status (*fastr.execution.linkrun.LinkRun* property), 182
- status (*fastr.execution.noderun.NodeRun* property), 190
- status (*fastr.planning.link.Link* property), 239
- status (*fastr.planning.node.Node* property), 250
- STATUS\_MAPPING (*fastr.plugins.SlurmExecution* attribute), 279
- STDERR\_DUMP (*fastr.execution.job.Job* attribute), 169
- stderrfile (*fastr.execution.job.Job* property), 172
- stderrurl (*fastr.execution.job.Job* property), 172
- stdout (in module *fastr.plugins*), 286
- STDOUT\_DUMP (*fastr.execution.job.Job* attribute), 169
- stdoutfile (*fastr.execution.job.Job* property), 173
- stdouturl (*fastr.execution.job.Job* property), 173
- store() (*fastr.helpers.filesynhelper.FileSyncHelper* method), 207
- StrongrExecution (class in *fastr.plugins*), 280
- SubInput (class in *fastr.planning.inputoutput*), 232
- SubInputRun (class in *fastr.execution.inputoutputrun*), 164
- submit\_jobs() (*fastr.plugins.DRMAAExecution* method), 258
- SubOutput (class in *fastr.planning.inputoutput*), 234
- SubOutputRun (class in *fastr.execution.inputoutputrun*), 166
- SubprocessBasedTarget (class in *fastr.core.target*), 126
- substitute() (*fastr.execution.job.SinkJob* method), 177
- suffix (*fastr.core.version.Version* property), 133
- SUPPORTED\_APIS (*fastr.plugins.PimReporter* attribute), 273
- SUPPORTS\_CANCEL (*fastr.plugins.DRMAAExecution* attribute), 256
- SUPPORTS\_CANCEL (*fastr.plugins.executionplugin.ExecutionPlugin* attribute), 288
- SUPPORTS\_CANCEL (*fastr.plugins.SlurmExecution* attribute), 279
- SUPPORTS\_DEPENDENCY  
(*fastr.plugins.DRMAAExecution* attribute), 256
- SUPPORTS\_DEPENDENCY  
(*fastr.plugins.executionplugin.ExecutionPlugin* attribute), 288
- SUPPORTS\_DEPENDENCY (*fastr.plugins.SlurmExecution* attribute), 279
- SUPPORTS\_HOLD\_RELEASE  
(*fastr.plugins.DRMAAExecution* attribute), 256
- SUPPORTS\_HOLD\_RELEASE  
(*fastr.plugins.executionplugin.ExecutionPlugin* attribute), 288
- SUPPORTS\_HOLD\_RELEASE  
(*fastr.plugins.SlurmExecution* attribute), 279
- swap() (*fastr.execution.environmentmodules.EnvironmentModules* method), 153
- switch\_sample\_sink() (in module *fastr.utils.cmd.trace*), 307
- sync() (*fastr.execution.environmentmodules.EnvironmentModules* method), 153
- systemdir (*fastr.helpers.configmanager.Config* property), 204
- SystemUsageInfo (class in *fastr.core.target*), 126
- ## T
- tags (*fastr.core.tool.Tool* attribute), 131
- Target (class in *fastr.core.target*), 127
- target (*fastr.core.tool.Tool* property), 131
- target (*fastr.datatypes.Deferred* property), 148
- target (*fastr.execution.linkrun.LinkRun* property), 182
- target (*fastr.planning.link.Link* property), 239

**TargetManager** (class in *fastr.plugins.managers.targetmanager*), 300  
**TargetResult** (class in *fastr.core.target*), 127  
**test()** (*fastr.core.interface.Interface* class method), 107  
**test()** (*fastr.core.target.Target* class method), 127  
**test()** (*fastr.core.tool.Tool* class method), 131  
**test()** (*fastr.datatypes.BaseDataType* class method), 142  
**test()** (*fastr.planning.network.Network* class method), 244  
**test()** (*fastr.plugins.BlockingExecution* class method), 254  
**test()** (*fastr.plugins.DRMAAExecution* class method), 258  
**test()** (*fastr.plugins.ElasticsearchReporter* class method), 260  
**test()** (*fastr.plugins.LinearExecution* class method), 267  
**test()** (*fastr.plugins.MacroTarget* class method), 270  
**test()** (*fastr.plugins.NipypeInterface* class method), 272  
**test()** (*fastr.plugins.ProcessPoolExecution* class method), 274  
**test()** (*fastr.plugins.RQExecution* class method), 275  
**test()** (*fastr.plugins.S3Filesystem* class method), 277  
**test()** (*fastr.plugins.SingularityTarget* class method), 278  
**test()** (*fastr.plugins.SlurmExecution* class method), 280  
**test()** (*fastr.plugins.StrongrExecution* class method), 280  
**test\_spec** (*fastr.core.tool.Tool* attribute), 131  
**test\_tool()** (*fastr.core.tool.Tool* class method), 131  
**time** (*fastr.api.ResourceLimit* property), 96  
**time** (*fastr.core.resourcelimit.ResourceLimit* property), 113  
**timestamp** (*fastr.core.target.SystemUsageInfo* property), 127  
**tmpdir** (*fastr.execution.job.Job* property), 173  
**tmpurl** (*fastr.execution.job.Job* property), 173  
**tmpurl** (*fastr.execution.job.SinkJob* property), 177  
**todict()** (*fastr.plugins.managers.objectmanager.ObjectManager* class method), 296  
**Tool** (class in *fastr.core.tool*), 128  
**tool** (*fastr.execution.job.Job* property), 173  
**tool** (*fastr.execution.noderun.NodeRun* property), 190  
**tool** (*fastr.planning.node.Node* property), 250  
**tool()** (in module *fastr.utils.cmd.test*), 307  
**TOOL\_REFERENCE\_FILE\_NAME** (*fastr.core.tool.Tool* attribute), 128  
**TOOL\_RESULT\_FILE\_NAME** (*fastr.core.tool.Tool* attribute), 128  
**toollist** (*fastr.utils.cmd.upgrade.FastrNamespaceType* property), 308  
**ToolManager** (class in *fastr.plugins.managers.toolmanager*), 300  
**tools** (*fastr* attribute), 77  
**tools\_path** (*fastr.helpers.configmanager.Config* property), 204  
**toolversions()** (*fastr.plugins.managers.toolmanager.ToolManager* class method), 301  
**TORQUE\_NATIVE\_SPEC** (*fastr.plugins.DRMAAExecution* attribute), 256  
**tostring\_modvalue()** (*fastr.execution.environmentmodules.EnvironmentModules* static method), 153  
**totuple\_modvalue()** (*fastr.execution.environmentmodules.EnvironmentModules* static method), 154  
**translate\_argument()** (*fastr.execution.job.Job* class method), 173  
**translate\_output\_results()** (*fastr.execution.job.Job* static method), 173  
**translate\_results()** (*fastr.execution.job.Job* class method), 173  
**TypeGroup** (class in *fastr.datatypes*), 149  
**typelist** (*fastr.utils.cmd.upgrade.FastrNamespaceType* property), 308  
**types** (*fastr* attribute), 78  
**types\_path** (*fastr.helpers.configmanager.Config* property), 204

## U

**unload()** (*fastr.execution.environmentmodules.EnvironmentModules* class method), 154  
**unmerge()** (*fastr.planning.inputgroupcombiner.BaseInputGroupCombiner* class method), 218  
**unmerge()** (*fastr.planning.inputgroupcombiner.DefaultInputGroupCombiner* class method), 219  
**unmerge()** (*fastr.planning.inputgroupcombiner.MergingInputGroupCombiner* class method), 220  
**unregister\_signals()** (*fastr.execution.networkrun.NetworkRun* class method), 187  
**update()** (*fastr.helpers.configmanager.EmptyDefault* class method), 205  
**update()** (*fastr.planning.inputgroupcombiner.BaseInputGroupCombiner* class method), 219  
**update()** (*fastr.planning.inputgroupcombiner.MergingInputGroupCombiner* class method), 220  
**update\_input\_groups()** (*fastr.execution.noderun.NodeRun* class method), 190  
**update\_input\_groups()** (*fastr.planning.node.Node* class method), 250  
**update\_size()** (*fastr.core.dimension.Dimension* class method), 104  
**upgrade\_network()** (in module *fastr.utils.cmd.upgrade*), 308  
**upgrade\_tool()** (in module *fastr.utils.cmd.upgrade*), 308

- `upload()` (*fastr.plugins.XNATStorage* static method), 286
  - `url` (*fastr.core.tool.Tool* attribute), 131
  - `url_to_path()` (*fastr.core.ioplugin.IOPlugin* method), 111
  - `url_to_path()` (*fastr.core.vfs.VirtualFileSystem* method), 136
  - `url_to_path()` (*fastr.plugins.FileSystem* method), 264
  - `url_to_path()` (*fastr.plugins.managers.iopluginmanager.IOPluginManager* method), 294
  - `URLType` (class in *fastr.datatypes*), 149
  - `usage_type` (*fastr.core.target.ProcessUsageCollection* attribute), 126
  - `userdir` (*fastr.helpers.configmanager.Config* property), 204
- ## V
- `VALID` (*fastr.core.samples.SampleState* attribute), 123
  - `valid` (*fastr.datatypes.BaseDataType* property), 142
  - `valid` (*fastr.datatypes.URLType* property), 150
  - `valid` (*fastr.execution.inputoutputrun.OutputRun* property), 162
  - `valid` (*fastr.execution.sourcenoderun.SourceNodeRun* property), 195
  - `valid` (*fastr.planning.inputoutput.Output* property), 231
  - `valid` (*fastr.planning.node.SourceNode* property), 253
  - `validate()` (*fastr.core.cardinality.CardinalitySpec* method), 100
  - `validate_results()` (*fastr.execution.job.Job* method), 173
  - `validate_results()` (*fastr.execution.job.SinkJob* method), 177
  - `validate_results()` (*fastr.execution.job.SourceJob* method), 179
  - `value` (*fastr.datatypes.BaseDataType* property), 142
  - `value` (*fastr.datatypes.Deferred* property), 148
  - `value` (*fastr.datatypes.Missing* attribute), 149
  - `ValueCardinalitySpec` (class in *fastr.core.cardinality*), 102
  - `ValueType` (class in *fastr.datatypes*), 150
  - `verify_resource_loading()` (in module *fastr.utils.verify*), 304
  - `verify_tool()` (in module *fastr.utils.verify*), 304
  - `verify_tool_instantiate()` (in module *fastr.utils.verify*), 304
  - `verify_tool_schema()` (in module *fastr.utils.verify*), 304
  - `Version` (class in *fastr.core.version*), 131
  - `version` (*fastr.api.Network* property), 81
  - `version` (*fastr.core.tool.Tool* attribute), 131
  - `version` (*fastr.datatypes.BaseDataType* attribute), 142
  - `version` (*fastr.datatypes.EnumType* attribute), 148
  - `version_matcher` (*fastr.core.version.Version* attribute), 133
  - `VirtualFileSystem` (class in *fastr.core.vfs*), 133
  - `VirtualFileSystem` (class in *fastr.plugins*), 280
  - `VirtualFileSystemRegularExpression` (class in *fastr.plugins*), 281
  - `VirtualFileSystemValueList` (class in *fastr.plugins*), 282
  - `vmem` (*fastr.core.target.SystemUsageInfo* property), 127
- ## W
- `wait_for_file()` (*fastr.helpers.filesynchelper.FileSyncHelper* method), 207
  - `wait_for_job()` (*fastr.helpers.filesynchelper.FileSyncHelper* method), 207
  - `wait_for_pickle()` (*fastr.helpers.filesynchelper.FileSyncHelper* method), 207
  - `wait_for_vfs_url()` (*fastr.helpers.filesynchelper.FileSyncHelper* method), 207
  - `warn_develop` (*fastr.helpers.configmanager.Config* property), 204
  - `web_hostname` (*fastr.helpers.configmanager.Config* property), 204
  - `web_url()` (*fastr.helpers.configmanager.Config* method), 204
  - `which()` (in module *fastr.helpers.procutils*), 212
  - `write()` (*fastr.execution.job.Job* method), 174
  - `write_bytes` (*fastr.core.target.SystemUsageInfo* property), 127
- ## X
- `xnat` (*fastr.plugins.XNATStorage* property), 286
  - `XNATStorage` (class in *fastr.plugins*), 283