

---

# **FASTR Documentation**

***Release 1.2.2***

**Marcel Koek and Hakim Achterberg**

**Sep 28, 2017**



---

## Contents

---

<b>1 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 . . . . .	9
1.3.1 Tools . . . . .	9
1.3.2 Network . . . . .	12
1.3.3 Data Flow . . . . .	14
1.3.4 DataTypes . . . . .	19
1.3.5 Execution . . . . .	20
1.3.6 IOPlugins . . . . .	21
1.3.7 Naming Convention . . . . .	21
1.3.8 Provenance . . . . .	21
1.4 Command Line Tools . . . . .	22
1.4.1 cat . . . . .	22
1.4.2 execute . . . . .	22
1.4.3 extract_argparse . . . . .	23
1.4.4 prov . . . . .	23
1.4.5 run . . . . .	23
1.4.6 testtool . . . . .	23
1.4.7 trace . . . . .	24
1.4.8 verify . . . . .	24
1.4.9 webapp . . . . .	24
1.5 Resource File Formats . . . . .	25
1.5.1 Config file . . . . .	25
1.5.2 Tool description . . . . .	27
1.6 Resource Reference . . . . .	29
1.6.1 CollectorPlugin Reference . . . . .	29
1.6.2 ExecutionPlugin Reference . . . . .	31
1.6.3 FlowPlugin Reference . . . . .	33
1.6.4 IOPlugin Reference . . . . .	33
1.6.5 Interface Reference . . . . .	38
1.7 Development and Design Documentation . . . . .	38
1.7.1 Sample flow in Fastr . . . . .	38
1.7.2 Network Execution . . . . .	41

1.8	Changelog . . . . .	43
1.8.1	1.2.2 - 2017-08-24 . . . . .	43
1.8.2	1.2.1 - 2017-04-04 . . . . .	43
1.8.3	1.2.0 - 2017-03-15 . . . . .	44
1.8.4	1.1.2 - 2016-12-22 . . . . .	45
1.8.5	1.1.1 - 2016-12-22 . . . . .	45
1.8.6	1.1.0 - 2016-12-08 . . . . .	45
<b>2</b>	<b>FASTR User reference</b>	<b>47</b>
2.1	Fastr User Reference . . . . .	47
<b>3</b>	<b>FASTR REST API reference</b>	<b>51</b>
3.1	REST API . . . . .	51
3.1.1	Quick reference . . . . .	51
<b>4</b>	<b>FASTR Developer Module reference</b>	<b>53</b>
4.1	fastr Package . . . . .	53
4.1.1	fastr Package . . . . .	53
4.1.2	configmanager Module . . . . .	64
4.1.3	datatypes Module . . . . .	66
4.1.4	exceptions Module . . . . .	75
4.1.5	plugins Module . . . . .	82
4.1.6	version Module . . . . .	95
4.1.7	Subpackages . . . . .	96
<b>5</b>	<b>Indices and tables</b>	<b>191</b>
	<b>HTTP Routing Table</b>	<b>193</b>
	<b>Python Module Index</b>	<b>195</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 bitbucket at [https://bitbucket.org/bigr\\_erasmusmc/fastr](https://bitbucket.org/bigr_erasmusmc/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.



# CHAPTER 1

---

## FASTR Documentation

---

### 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).

### 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.

### 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:

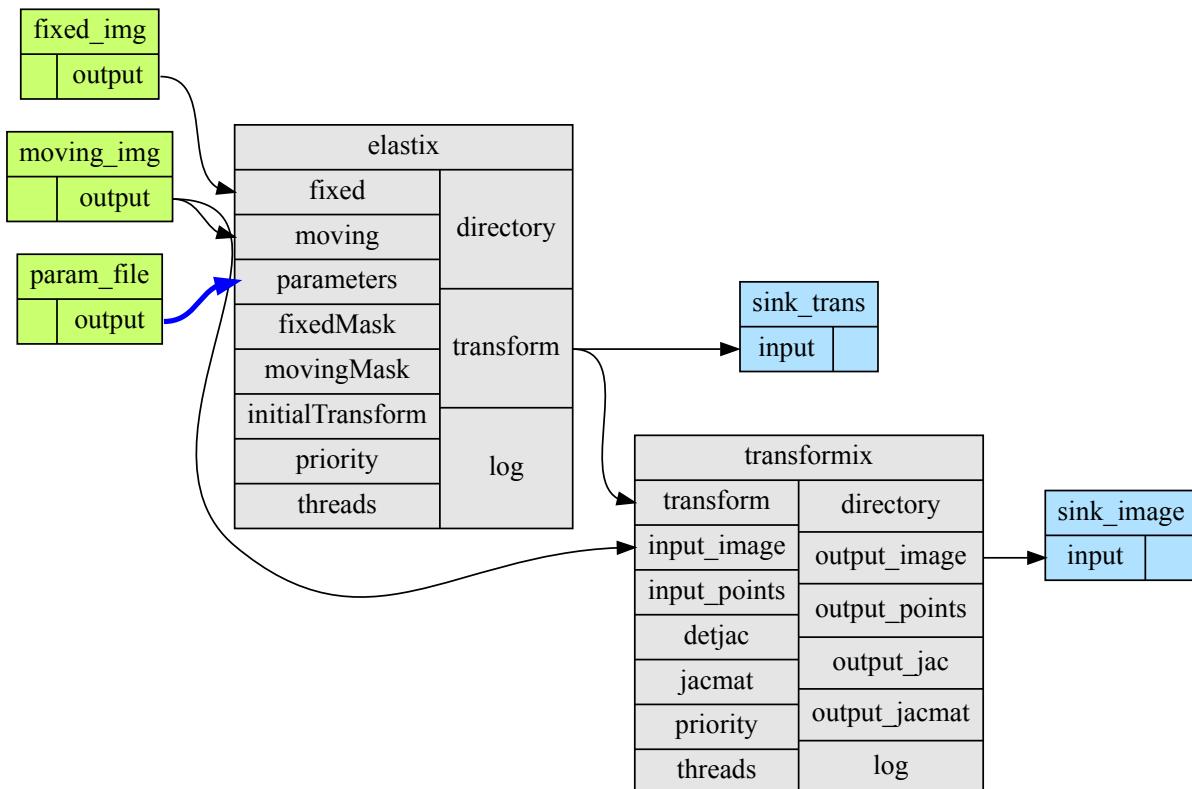


Fig. 1.1: A simple workflow that registers two images and uses the resulting transform to resample the moving image.

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.

## Quick start guide

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

### 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:

```
hg clone https://<yourusername>@bitbucket.org/bigr_erasmusmc/fastr # for http
hg clone ssh://hg@bitbucket.org/bigr_erasmusmc/fastr # for ssh
```

If you prefer a GUI you can try [TortoiseHG](#) (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://<yourusername>@bitbucket.org/bigr_erasmusmc/fastr
ssh://hg@bitbucket.org/bigr_erasmusmc/fastr
```

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/python2.7/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`.
- 

## 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` and `~/.fastr/` directories. If both config files contain values for a single setting, the version in `~/.fastr/` has priority.

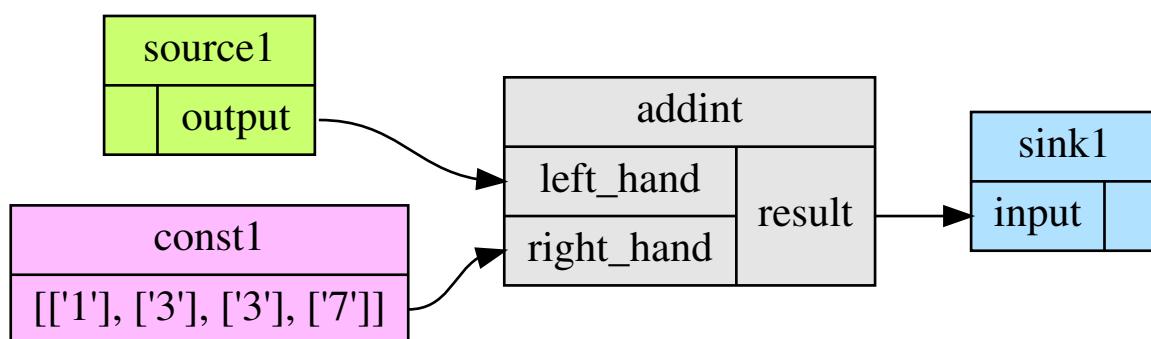
For a sample configuration file and a complete overview of the options in `config.py` see the [Config file](#) section.

## 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.Network()
```

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. The easiest way to do this is via the network `create_` methods. Let's create two source nodes, one normal node, and one sink:

```
>>> source1 = network.create_source('Int', id_='source1')
>>> constant1 = network.create_constant('Int', [1, 3, 3, 7], id_='const1')
>>> sink1 = network.create_sink('Int', id_='sink1')
>>> addint = network.create_node('AddInt', id_='addint')
```

The functions `Network.create_source`, `Network.create_constant`, `Network.create_sink` and `Network.create_node` are shortcut functions for calling the `SourceNode`, `ConstantNode`, `SinkNode` and `Node` constructors and adding them to the network. A `SourceNode` and `SinkNode` only require the datatype to be specified. A `ConstantNode` requires both the datatype and the data to be set on creation. A `Node` requires a `Tool` template to be instantiated from. The `id_` option is optional for all three, but makes it easier to identify the nodes and read the logs.

There is an easier 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 contant and the input:

```
>>> addint.inputs['right_hand'] = [1, 3, 3, 7]
```

The created constant would have the id `addint_right_hand_const` as it automatically names the new constant `$nodeid$inputid_const`.

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 v1.0)
    Inputs           |       Outputs
    -----
                    |   output   (Int)

>>> addint
Node addint (tool: AddInt 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.toollist` and `fastr.typeplist`. These variables are created when `fastr` is imported for the first time. They contain all the datatype and tools specified by the xml files in the search paths. To get an overview of the tools and datatypes loaded by `fastr`:

```
>>> fastr.toollist
ToolManager
Add                         v0.1          : /home/hachterberg/dev/fastr-
→develop/fastr/fastr/resources/tools/add/v1_0/add.xml
AddImages                   v0.1          : /home/hachterberg/dev/fastr-
→develop/fastr/fastr/resources/tools/addimages/v1_0/addimages.xml
AddInt                      v0.1          : /home/hachterberg/dev/fastr-
→develop/fastr/fastr/resources/tools/addint/v1_0/addint.xml

>>> fastr.typeplist
DataTypeManager
AnyType                     : <class 'fastr.datatypes.AnyType'>
Boolean                     : <class 'fastr.datatypes.Boolean'>
Directory                   : <class 'fastr.datatypes.Directory'>
Float                       : <class 'fastr.datatypes.Float'>
```

```
Int : <class 'fastr.datatypes.Int'>
String : <class 'fastr.datatypes.String'>
```

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

- `toolist[id]` which returns the newest version of the tool
- `toolist[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 = network.create_link(source1.output, addint.inputs['left_hand'])
>>> link2 = network.create_link(constant1.output, addint.inputs['right_hand'])
>>> link3 = network.create_link(addint.outputs['result'], sink1.inputs['input'])
```

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 a link. If not, it is possible to use a short-hand which creates a link but gives you no easy access to it for later.

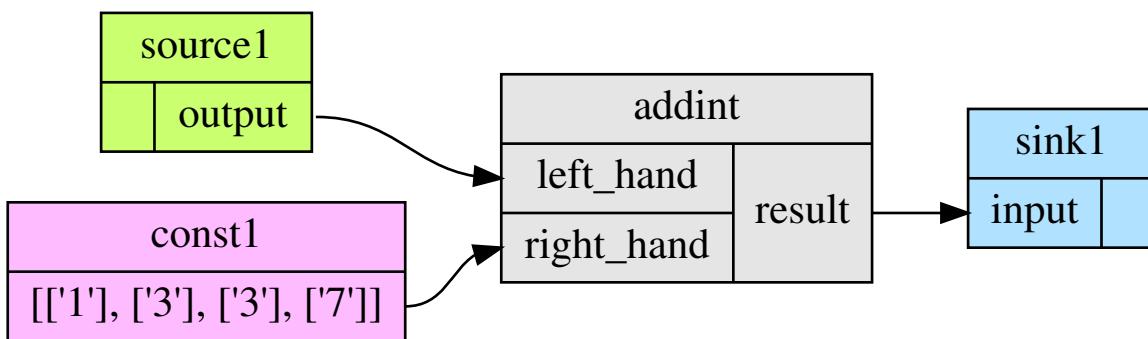
```
>>> addint.inputs['left_hand'] = source1.output
>>> addint.inputs['right_hand'] = constant1.output
>>> sink1.inputs['input'] = addint.outputs['result']
```

### 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_network` method.

```
>>> network.draw_network()
'/home/username/network_layout.dot.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

## 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'}
>>> network.execute(source_data, sink_data)
```

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-lanuage (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](#)

## 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.

### Tools

The `Tools` 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`. The 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.toollist`. To get an overview of the tools in the system, just print the `repr()` of the `ToolManager`:

```
>>> fastr.toollist
AddImages           v0.1      : /home/hachterberg/dev/fastr/fastr/resources/
  ↳ tools/addimages/v1_0/addimages.xml
AddInt              v0.1      : /home/hachterberg/dev/fastr/fastr/resources/
  ↳ tools/addint/v1_0/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.toollist['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
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.

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" nospace="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>
```

```
</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 toollist when you type fastr.toollist.
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 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.toollist
```

Now a list of available tools should be produced, including the tool throw\_die

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

```
import fastr
network = fastr.Network()
source1 = network.create_source(fastr.typeList['Int'], id_='source1')
sink1 = network.create_sink(fastr.typeList['Int'], id_='sink1')
throwdie = network.create_node(fastr.toolList['ThrowDie'], id_='throwdie')
link1 = network.create_link(source1.output, throwdie.inputs['die_sides'])
link2 = network.create_link(throwdie.outputs['output'], sink1.inputs['input'])
source_data = {'source1': {'s1': 4, 's2': 5, 's3': 6, 's4': 7}}
sink_data = {'sink1': 'vfs://tmp/fastr_result_{sample_id}.txt'}
network.draw_network()
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">
    <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">
```

```

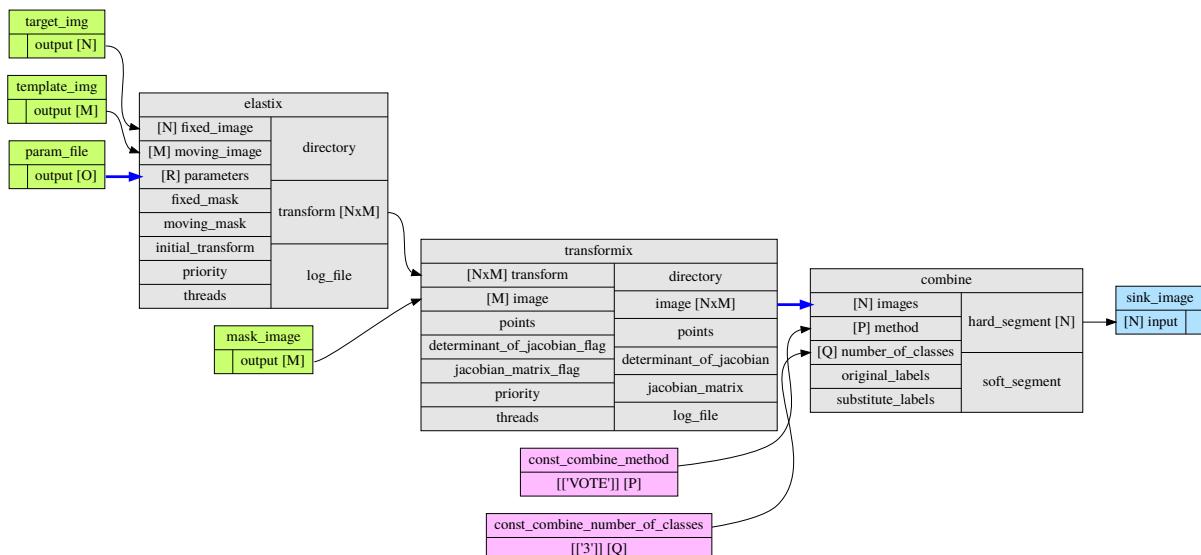
<paths>
  <path type="bin" value="vfs://apps/elastix/4.7/install/" />
  <path type="lib" value="vfs://apps/elastix/4.7/install/lib" />
</paths>
</target>
<target os="darwin" arch="*" modules="elastix/4.7" bin="elastix">
  <paths>
    <path type="bin" value="vfs://apps/elastix/4.7/install/" />
    <path 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](#).

## 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.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 in a two different ways:

```
>>> node1 = fastr.Node(tool, id_='node1', parent=network)
>>> node2 = network.create_node(tool, id_='node2', stepid='step1')
```

In the first way, we specifically create a *Node* object. We pass it an `id` and the parent *network*. If the `parent` is `None` the `fastr.current_network` will be used. The *Node* constructor will automatically add the new node to the parent *network*.

---

**Note:** For a Node, the tool can be given both as the `Tool` class or the id of the

tool.

The second way, we tell the network to create a `Node`. The network will automatically assign itself as the parent. Optionally you can add define a stepid for the node which is a logical grouping of `Nodes` that is mostly used for visualization.

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

```
>>> addint = fastr.Node(fastr.toollist['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
InputDict([('left_hand', <Input: fastr:///networks/unnamed_network/nodelist/addint/
    ↪inputs/left_hand>), ('right_hand', <Input: fastr:///networks/unnamed_network/
    ↪nodelist/addint/inputs/right_hand>) ])

>>> addint.outputs
OutputDict([('result', Output fastr:///networks/unnamed_network/nodelist/addint/
    ↪outputs/result)])
```

The `InputDict` and `OutputDict` are classes that behave like mappings. The `InputDict` also facilitates the linking shorthand. By assigning an `Output` to an existing key, the `InputDict` will create a `Link` between the `InputDict` 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 `IOPplugins`. 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 = fastr.SourceNode('Int', id_='source1')
>>> source2 = network.create_source(fastr.typelist['Int'], id_='source2', stepid=
    ↪'step1')
```

In both cases, the source is automatically automatically assigned to a network. In the first case to the `fastr.current_network` and in the second case to the network used to call the method. 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 = fastr.ConstantNode('Int', [42], id_='constant1')
>>> constant2 = network.create_constant('Int', [42], id_='constant2', stepid='step1
→')
```

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:

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

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:

```
>>> sink1 = fastr.Sink('Int', id_='sink1')
>>> sink2 = network.create_sink(fastr.typeplist['Int'], id_='sink2', stepid='step1')
```

## Link

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

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

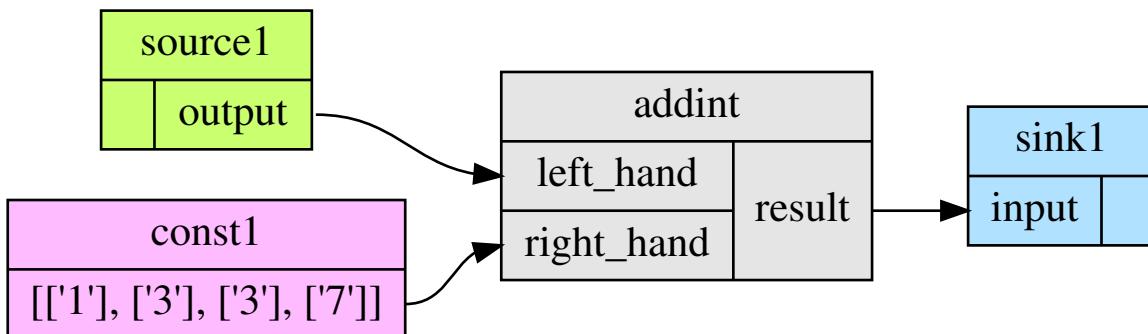
or can be created implicitly by assigning an `Output` to an `Input` in the `InputDict`.

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

Note that a `Link` is also created automatically when using the short-hand for the `ConstantNode`

## Data Flow

The data enters the `Network` via `SourceNodes`, flows via other `Nodes` 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 (possibly while transforming the data).

### Data flow inside a Node

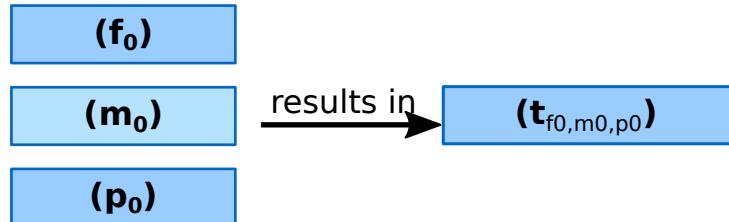
In a *Node* all data from the *Inputs* will be combined and the jobs will be generated. There are strict rules to how this combination is performed. In the default case all inputs will be used pair-wise, and if there is only a single value for an input, it will be considered as a constant.

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

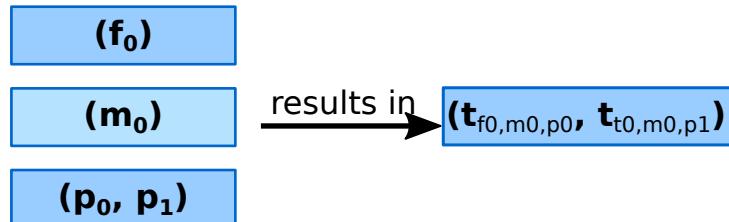
>>> fastr.toollist['Elastix']		
Tool Elastix v4.8 (Elastix Registration)		
		Inputs   Outputs
<hr/>		
fixed_image	(ITKImageFile)	transform
↳ (ElastixTransformFile)		
moving_image	(ITKImageFile)	
parameters	(ElastixParameterFile)	

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 Inputs*.

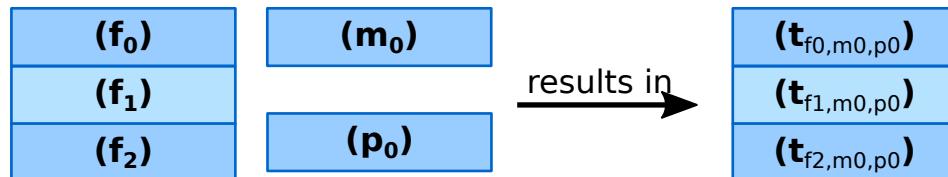
If we supply a *Node* based on this *Tool* with a single sample on each *Input*, there will be one single matching *Output* sample created:



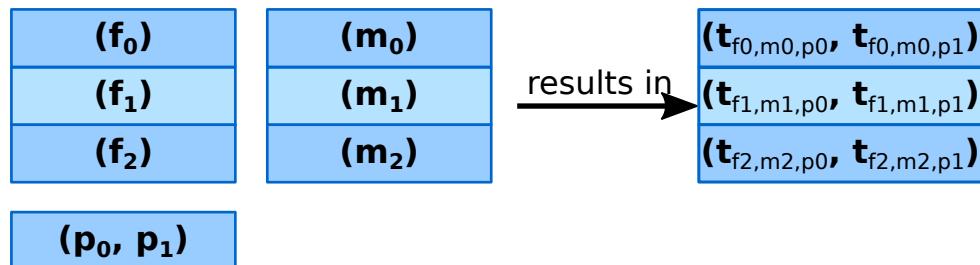
If the cardinality of the *parameters* sample would be increased to 2, the resulting *transform* sample would also become 2:



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.

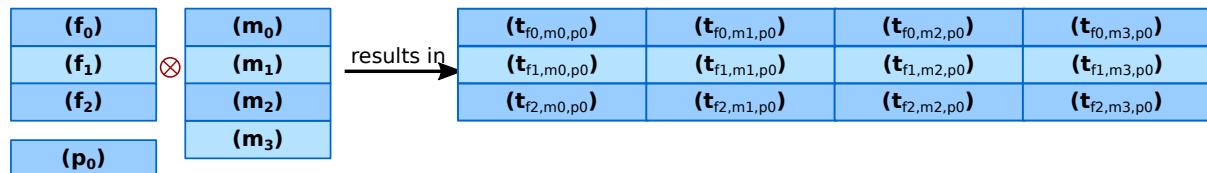


### Advanced flows in a Node

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:

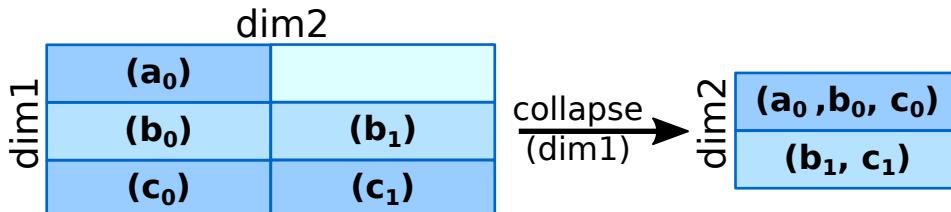


**Warning:** TODO: Expand this section with the merging dimensions

### Data flows in a Link

As mentioned before the data flows from an `Output` to an `Input` through a `Link`. By default the `Link` passes 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.

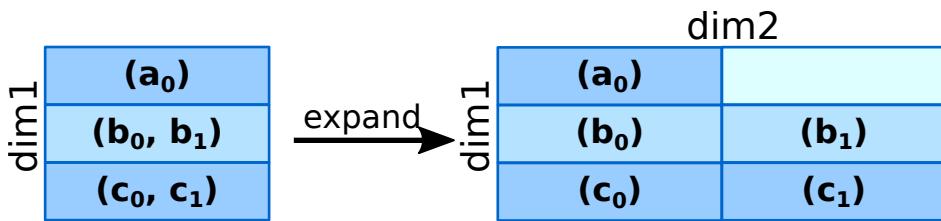


This is useful in situation where you want to use a tool that aggregates over a number of samples (e.g. take a mean or sum).

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}`



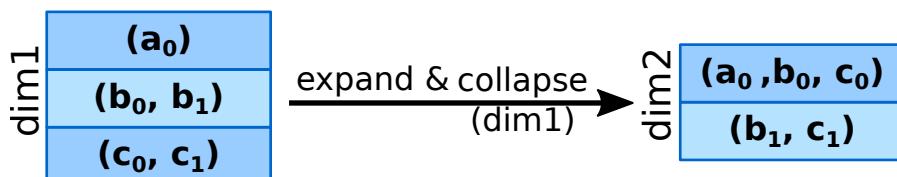
This flow directive is useful if you want to split a large sample in multiple smaller samples. This could be because processing the whole sample is not feasible because of resource constraints. An example would be splitting a 3D image into slices to process separately to avoid high memory use or to achieve 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 `Inputs` has multiple `Links` attached to it, the data will be combined by concatenating the values for each corresponding sample in the cardinality.

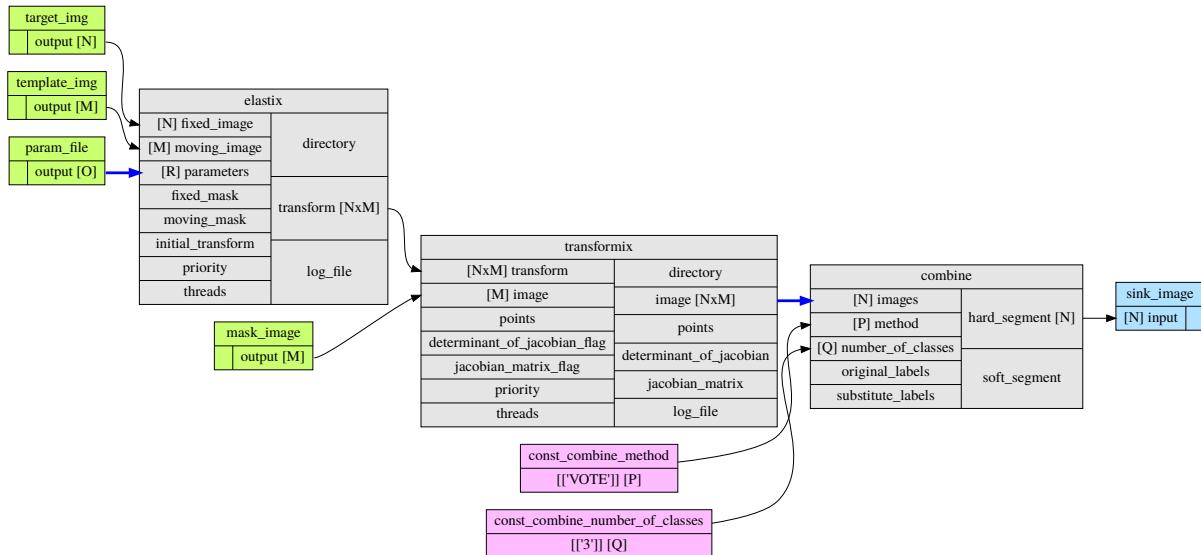
## Broadcasting (matching data of different dimensions)

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:

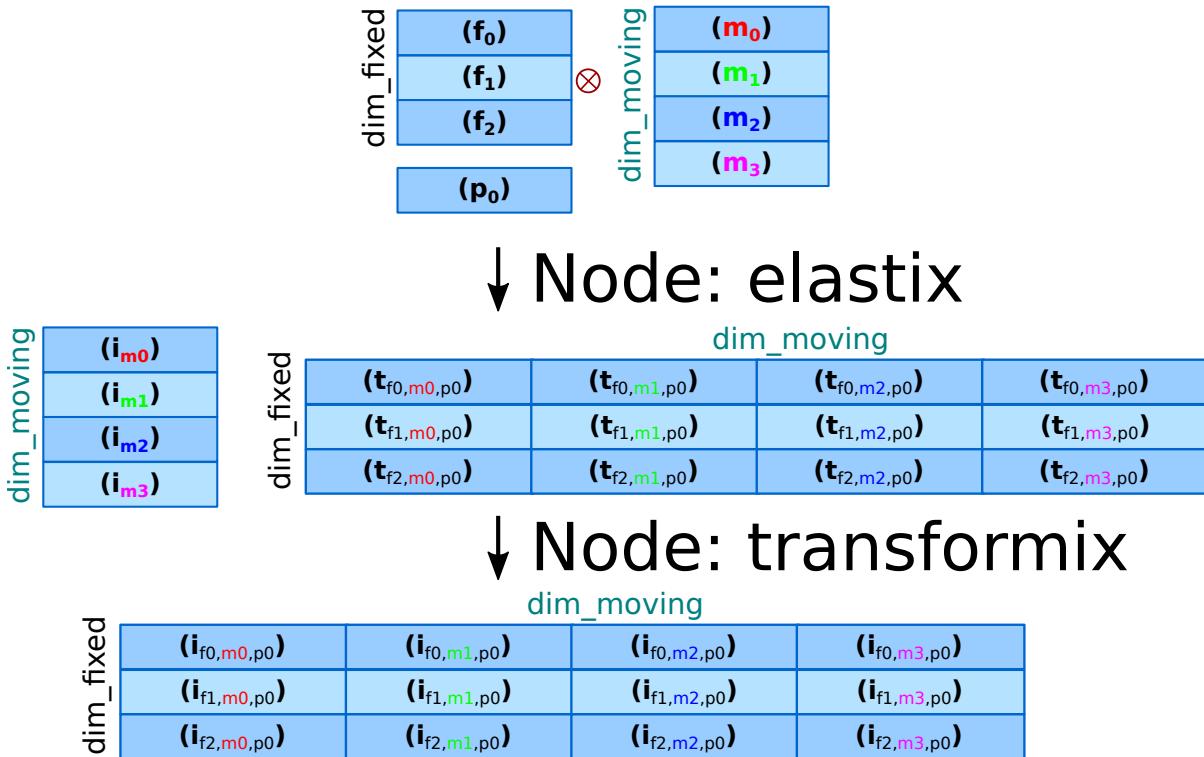
“The term broadcasting describes how numpy treats arrays with different shapes during arithmetic operations. Subject to certain constraints, the smaller array is “broadcast” across the larger array so that they have compatible shapes.”

—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 x 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 x M]`, it is possible to match them uniquely. This is done automatically and the result is a new `[N x M]` sample collection. To create a matching sample collections, the samples in the `transformix.image`

Inputs 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!

## DataTypes

In Fastr all data is contained in objects 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

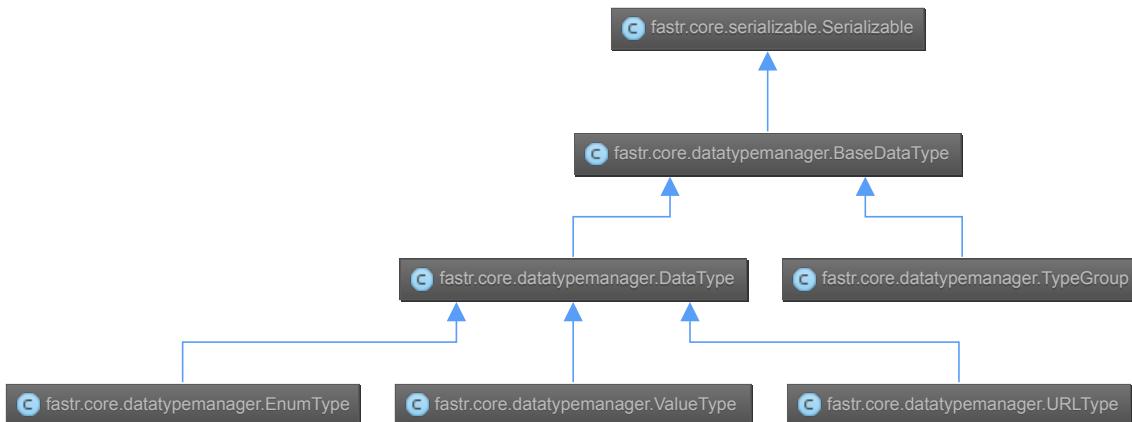


Fig. 1.2: The relation between the different `DataType` classes

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.typeList`. In Fastr the created `DataType` classes are also automatically placed in the `fastr.datatypes` module once created.

## Resolving Datatypes

`Outputs` in Fastr can have a `TypeGroup` or a number of `DataType`s 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 `DataType`s 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 `DataType`s (*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*

## 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*.

TODO: add .. figure::: images/execution\_layers.\*

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.

## IOPplugins

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 handled by the `VirtualFileSystem` plugin. A list of all the *IOPplugins* known by the system and their use can be found at [IOPPlugin Reference](#).

## 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 be 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	<code>module</code>	<code>brain_tissue_segmentation</code>
Tool.id	<code>class</code>	<code>BrainExtractionTool, ThresholdImage</code>
Node.id	<code>variable name</code>	<code>brain_extraction, threshold_mask</code>
Input/Output.id	<code>method</code>	<code>image, number_of_classes, probability_image</code>

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.

## 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>

<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>“

## 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 visualized using the `fastr prov` command line tool.

## Command Line Tools

command	description
<code>cat</code>	Print information from a job file
<code>execute</code>	Execute a fastr job file
<code>extract_argparse</code>	Create a stub for a Tool based on a python script using argparse
<code>prov</code>	Get PROV information from the result pickle.
<code>run</code>	Run a Network from the commandline
<code>testtool</code>	Run the tests of a tool to verify the proper function
<code>trace</code>	Trace samples/sinks from a run
<code>verify</code>	Print information from a job file
<code>webapp</code>	Start the fastr webapp and open in a new browser tab

### cat

```
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

### execute

Execute a job or network from commandline

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

#### Positional Arguments

**JOBFILE**

File of the job to execute (default ./\_\_fastr\_command\_\_.pickle.gz)

## extract\_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

## prov

Get PROV information from the result pickle. When no options are given, the provenance syntax is printed to stdout in PROV-JSON format.

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

### Positional Arguments

<b>RESULTFILE</b>	File of the job to execute (default ./__fastr_result__.pickle.gz)
-------------------	---

### Named Arguments

<b>-so, --syntax-out-file</b>	Write the syntax to file.
<b>-sf, --syntax-format</b>	Choices are: [json], provn or xml Default: “json”
<b>-i, --indent</b>	Indent size of the serialized documents. Default: 2
<b>-vo, --visualize-out-file</b>	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.

## run

Execute a job or network from commandline

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

### Positional Arguments

<b>NETWORKFILE</b>	File of the network to execute
--------------------	--------------------------------

## testtool

Run the tests for a Tool to check the function

```
usage: fastr testtool [-h] TOOL
```

## Positional Arguments

**TOOL** the id of the tool to test

## trace

```
usage: fastr trace [-h] [--verbose] [--sinks [SINKS [SINKS ...]]]
                   [--nodes [NODES [NODES ...]]]
                   [--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/1.2.2/fastr/doc/\_\_sink\_d

## Named Arguments

**--verbose, -v** path of the data to print

Default: False

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

**--nodes** list results for specified nodes

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

## verify

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

## Positional Arguments

**TYPE** Possible choices: tool

Type of resource to verify (e.g. tool)

**path** path of the resource to verify

## webapp

Fastr web client

```
usage: fastr webapp [-h] [-d] [-o]
```

## Named Arguments

**-d, --debug** Debug mode.

Default: False

**-o, --openpage** Open web page after start.

Default: False

## 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.

### Config file

Fastr reads the config files from the following locations by default (in order):

- \$FASTRHOME/config.py
- ~/.fastr/config.py

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.

### 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
logdir	str	Directory where the fastr logs will be placed	\$userdir/logs
logtype	str	Type of logging to use	'default'
mounts	dict	A dictionary containing all mount points in the VFS system	{'tmp': '\$TMPDIR', 'home': '~', 'example_data': '\$systemdir/examples/data'}
networks_path	list	Directories to scan for networks	[\$userdir/networks', '\$resourcedir/networks']
pim_host	str	Host of the PIM server to report to	"
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 environment modules that are protected against unloading	[]
resources-dir	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	4
systemdir	str	Fastr installation directory	'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python3.7/site-packages/fastr-1.2.2-py2.7.egg/fastr'
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	~/fastr
warn_develbp	bool	Warning users on import if this is not a production version of fastr	True
web_hostname	str	The hostname to expose the web app for	'localhost'
web_port	str	The port to expose the web app on	'5000'
web_secretskey	str	The secret key to use for the flask web app	'VERYSECRETKEY!'

## 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

Table 1.1 – continued from previous page

Attribute	Description	
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 targetted (windows, linux, macos or * (for any)
		arch Architecture targetted 32, 64 or * (for any)
		module Environment module giving access to the Tool
	location	If the module is not found, try using this location to find the Tool
		interpreter Interpreter to use to call the bin with (e.g. bash, python, Rscript)
		bin Name of the Tool binary (e.g. toolname, toolname.exe, toolname.py)
	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
inputs []		List of Inputs that can be 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>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>1</sup>
	enum []	List of possible values for an EnumType (created on the fly by fastr) <sup>1</sup>
	prefix	Commandline prefix of the Output (e.g. -out, -o)
	cardinality	Cardinality of the Output
	repeat_prefix	Flag indicating if for every value of the Output the prefix is repeated
	required	Flag indicating if the output is required
	nospace	Flag indicating if there is no space between prefix and value (e.g. -out=val)
	format	For DataTypes that have multiple representations, indicate which one to use
	description	Long description for an output
	action	Special action (defined per DataType) that needs to be performed before creation
	automatic	Indicate that output doesn't require commandline argument, but is created automatically
	method	Method to acquire output value from the Tool can be 'path' or 'stdout' <sup>2</sup>
	location	Definition where to an automatically, usage depends on the method <sup>2</sup>
help		Help text explaining the use of the Tool
cite		Bibtext of the Citation(s) to reference when using this Tool for a publication

<sup>1</sup> datatype and enum are conflicting entries, if both specified datatype has precedence<sup>2</sup> More details on defining automatica output are given in [TODO]

## Resource 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.

### 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

---

## 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 traverse the dependency graph and kill each job manual.

---

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

<code>scheme</code>	<code>ExecutionPlugin</code>
<code>BlockingExecution</code>	<code>BlockingExecution</code>
<code>DRMAAExecution</code>	<code>DRMAAExecution</code>
<code>LinearExecution</code>	<code>LinearExecution</code>
<code>ProcessPoolExecution</code>	<code>ProcessPoolExecution</code>
<code>RQExecution</code>	<code>RQExecution</code>

## 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

name	type	description	default
drmaa_queue	str	The default queue to use for jobs send to the scheduler	'week'

## 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

name	type	description	default
process_pool_worker_number	int	Number of workers to use in a process pool	3

## 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

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

## 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	<a href="#">CrossValidation</a>

### 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.

## 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	<a href="#">CommaSeperatedValueFile</a>
FileSystem	<a href="#">FileSystem</a>
Null	<a href="#">Null</a>
Reference	<a href="#">Reference</a>
VirtualFileSystem	<a href="#">VirtualFileSystem</a>
VirtualFileSystemRegularExpression	<a href="#">VirtualFileSystemRegularExpression</a>
VirtualFileSystemValueList	<a href="#">VirtualFileSystemValueList</a>
XNATStorage	<a href="#">XNATStorage</a>

### 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 true or false (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 value = reformat.format(value) (used before relative_path)
relative_path	indicates the entries are relative paths (for files), can be true or false (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:/

## 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.

## 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 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 at 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: project, subject, experiment, scan, or resource
label	false	Indicate the XNAT label should be used as fastr id, options true or false
insecure	false	Change the url scheme to be used to http instead of https
regex	false	Change search to use regex <code>re.match()</code> instead of fnmatch for matching

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.

## Interface Reference

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

<b>scheme</b>	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.

### 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

---

### 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.

---

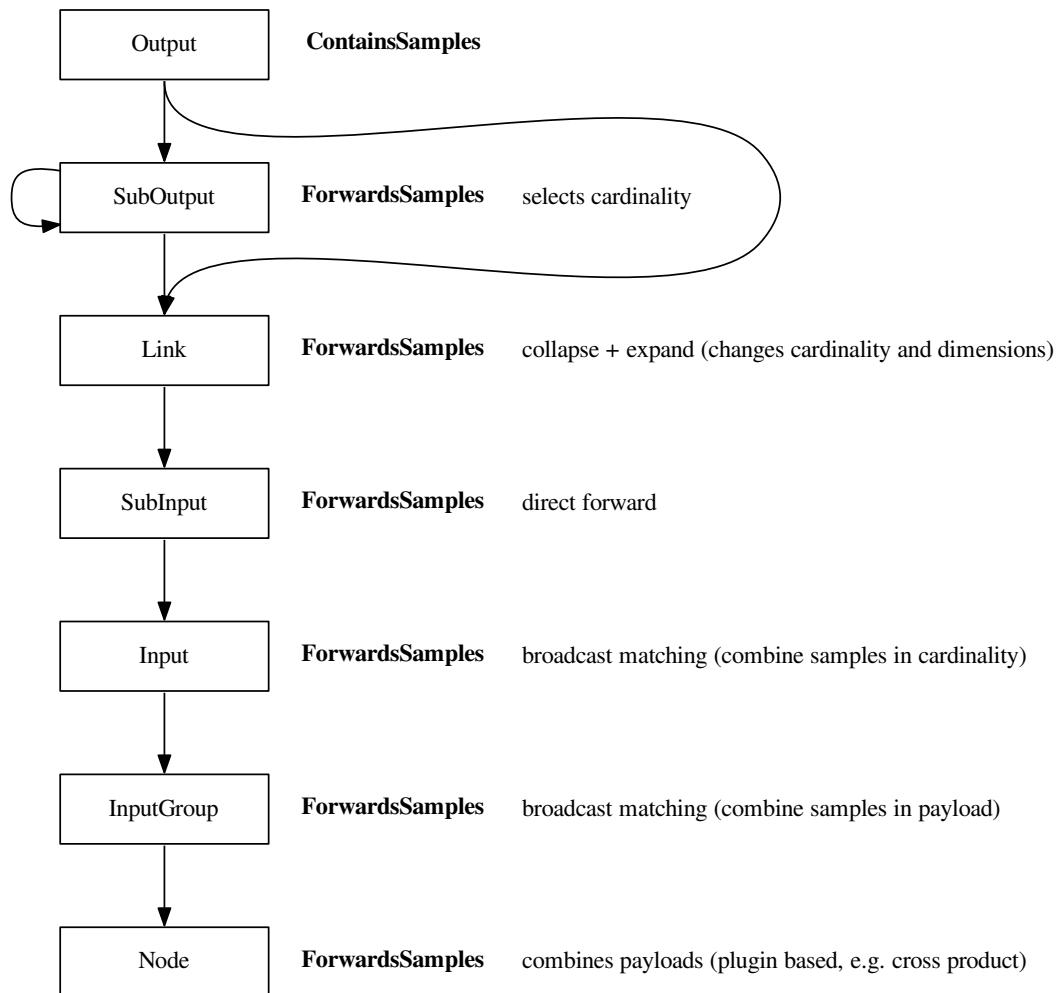
**Warning:** This interface plugin is basically functional, but highly experimental!

## Development and Design Documentation

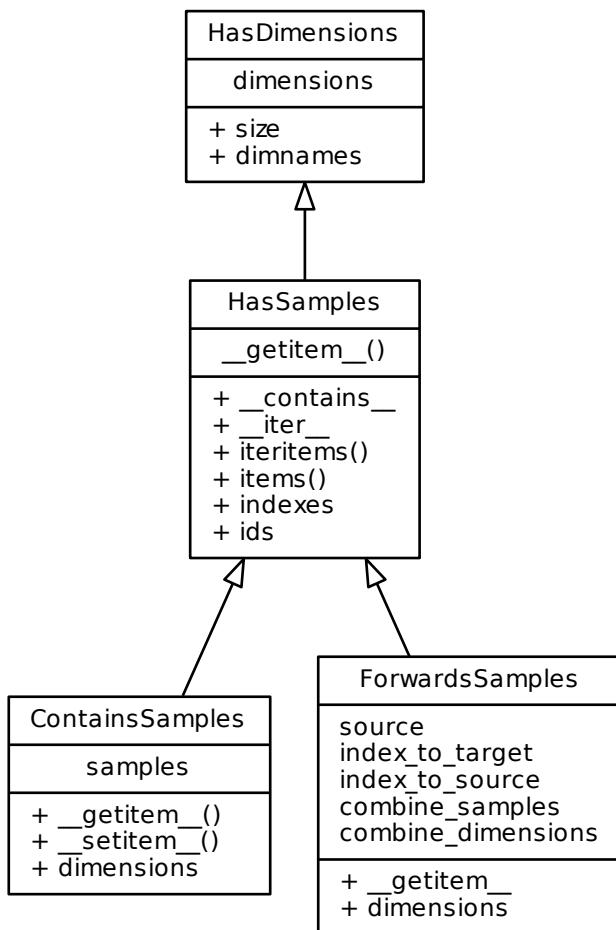
In this chapter we will discuss the design of Fastr in more detail. We give pointers for development and add the design documents as we currently envision Fastr. This is both for people who are interested in the Fastr develop and for current developers to have an archive of the design decision agreed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

### 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

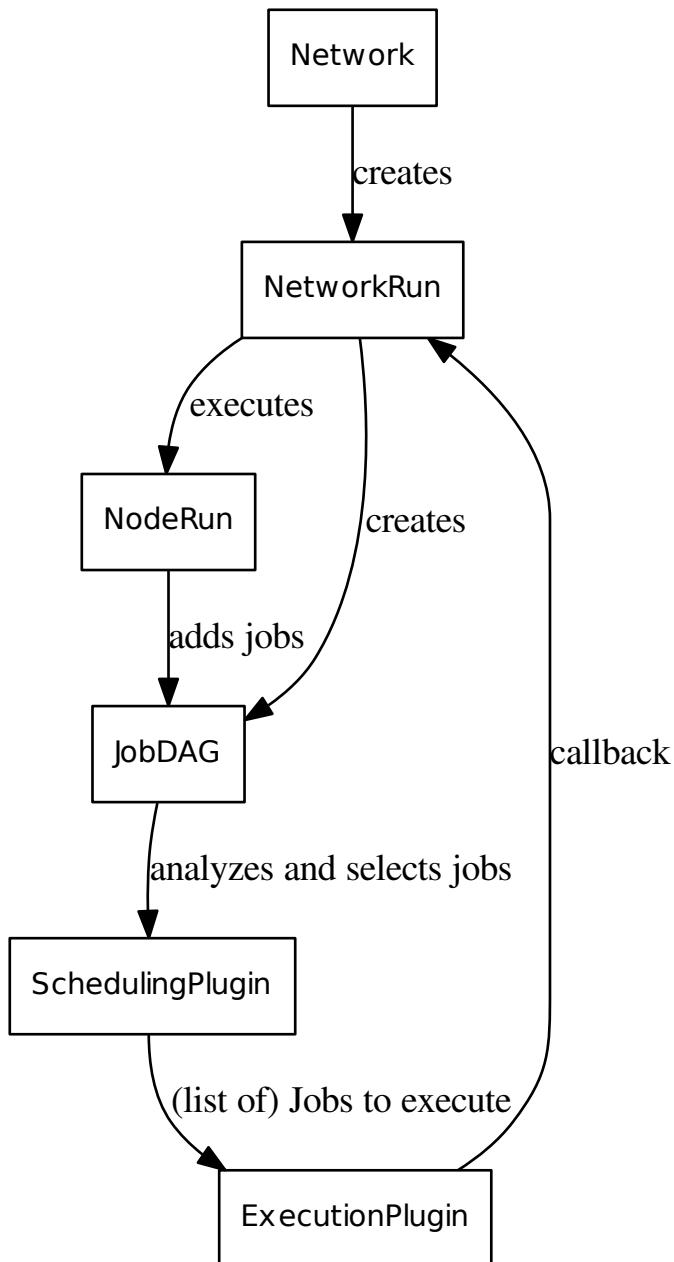
**Note:** Though the flow is currently working like this, the mixins are not yet created.

## 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 Node 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 multiple 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 independent of each other. Also editting 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 processed 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 Source. 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 Node when all samples are done with the previous Node, 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 limit the number of jobs an external scheduler has to track.

---

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

---

## 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](#)

### 1.2.2 - 2017-08-24

#### Fixed

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

### 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.

**1.2.0 - 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

### 1.1.2 - 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.

### 1.1.1 - 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.

### 1.1.0 - 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.

# CHAPTER 2

## FASTR User reference

### Fastr User Reference

#### fastr.toollist

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.toollist['elastix.Elastix']
Tool Elastix v4.8 (Elastix Registration)
    Inputs
    |
    ↵Outputs
    -----
    ↵fixed_image      (ITKImageFile)           | directory
    ↵(Directory)          | transform
    ↵moving_image     (ITKImageFile)           | log_file
    ↵(ElastixTransformFile)          |
    ↵parameters       (ElastixParameterFile) |
    ↵(ElastixLogFile)          |
    ↵fixed_mask        (ITKImageFile)           |
    ↵moving_mask       (ITKImageFile)           |
    ↵initial_transform (ElastixTransformFile) |
    ↵priority         (__Elastix_4.8_interface_priority__Enum__)
    ↵threads          (Int)                  |

>>> fastr.toollist['elastix.Elastix', '4.5']
Tool Elastix v4.5 (Elastix Registration)
    Inputs
    |
    ↵Outputs
    -----
    ↵fixed_image      (ITKImageFile)           | directory
    ↵(Directory)          | transform
    ↵moving_image     (ITKImageFile)           | log_file
    ↵(ElastixTransformFile)          |
    ↵parameters       (ElastixParameterFile) |
    ↵(ElastixLogFile)          |
    ↵fixed_mask        (ITKImageFile)           |
```

moving_mask	(ITKImageFile)
initial_transform	(ElastixTransformFile)
priority	(__Elastix_4.5_interface_priority_Enum__)
threads	(Int)

**fastr.typeList**

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

**class fastr.Network (id\_=’unnamed\_network’, version=None)**

The Network class represents a workflow. This includes all Nodes (including ConstantNodes, SourceNodes and Sinks) and Links.

**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

**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\_node (tool, id\_=None, stepid=None, cores=None, memory=None, walltime=None, node\_group=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
- **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\_sink (datatype, id\_=None, stepid=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

**Returns** the newly created sink node

**Return type** `SinkNode`

**create\_source** (`datatype`, `id_=None`, `stepid=None`, `nodegroup=None`, `sourcegroup=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
- **nodegroup** (`str`) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.
- **sourcegroup** (`str`) – *DEPRECATED!* The nodegroup this `SourceNode` will be added to

**Returns** the newly created source `source_node`

**Return type** `SourceNode`

**draw\_network** (`name='network_layout'`, `img_format='svg'`, `draw_dimension=False`)

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`

**class** `fastr.Link` (`source`, `target`, `parent=None`, `id_=None`, `collapse=None`, `expand=None`)

Class for linking outputs (`BaseOutput`) to inputs (`BaseInput`)

Examples:

```
>>> import fastr
>>> network = fastr.Network()
>>> link1 = network.create_link( n1.outputs['out1'], n2.inputs['in2'] )

link2 = Link()
link2.source = n1.outputs['out1']
link2.target = n2.inputs['in2']
```

#### 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`

#### 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`

**class** `fastr.Node` (`tool`, `id_=None`, `parent=None`, `cores=None`, `memory=None`, `walltime=None`)

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.

**id\_ = None**

The Node id is a unique string identifying the Node

**inputs = None**

A list of inputs of this Node

**name**

Name of the Tool the Node was based on. In case a Toolless Node was used the class name is given.

**outputs = None**

A list of outputs of this Node

**class fastr.ConstantNode (datatype, data, id\_=None)**

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.

**name**

Name of the Tool the Node was based on. In case a Toolless Node was used the class name is given.

**class fastr.SourceNode (datatype, id\_=None)**

Class providing a connection to data resources. This can be any kind of file, stream, database, etc from which data can be received.

**name**

Name of the Tool the Node was based on. In case a Toolless Node was used the class name is given.

**class fastr.SinkNode (datatype, id\_=None)**

Class which handles where the output goes. This can be any kind of file, e.g. image files, textfiles, config files, etc.

**name**

Name of the Tool the Node was based on. In case a Toolless Node was used the class name is given.

# CHAPTER 3

---

## FASTR REST API reference

---

### REST API

Fastr includes a webapp and a REST API. This section contains a listing of all REST paths. The full documentation is available via swagger with the paths /swagger.json for the schema or /api/doc/ for the swagger UI.

#### Quick reference

Resource	Operation	Description
	<i>GET /</i>	
	<i>GET /api/doc/</i>	
	<i>GET /api/networks</i>	
	<i>GET /api/networks/(id)</i>	
	<i>POST /api/runs</i>	
	<i>GET /api/runs</i>	
	<i>DELETE /api/runs/(id)</i>	
	<i>GET /api/runs/(id)</i>	
	<i>GET /api/runs/(id)/status</i>	
	<i>GET /api/tools</i>	
	<i>GET /api/tools/(id)</i>	
	<i>GET /api/tools/(id)/(version)</i>	
	<i>GET /doc</i>	
	<i>GET /index</i>	
	<i>GET /</i>	
	<i>GET /network/(name)</i>	
	<i>GET /networks</i>	
	<i>GET /prov</i>	
	<i>GET /shutdown</i>	
	<i>GET /static/(path:filename)</i>	
	<i>GET /swagger.json</i>	
	<i>GET /swaggerui/(path:filename)</i>	
	<i>GET /tool/(toolname)/(version)</i>	
	<i>GET /tool/(toolname)</i>	
	<i>GET /tool</i>	
	<i>GET /websocketclient</i>	

**GET /api/networks**

Get a list of the networks

**GET /api/tools**

Get a list of all Tools known to the server

**POST /api/runs**

Create a new Network run and start execution

**GET /api/runs**

Get a list of all Network runs on the server

**GET /api/doc/**

Override this method to customize the documentation page

**GET /swagger.json**

Render the Swagger specifications as JSON

**GET /api/tools/ (id) /**

*version* Get a Tool json description from the server

**GET /api/runs/ (id) /status**

Get the status of a Network Run on the server

**GET /api/networks/ (id)**

Get a Network json description from the server

**GET /api/tools/ (id)**

Get a Tool json description from the server

**DELETE /api/runs/ (id)**

Abort a Network run and stop all associated execution

**GET /api/runs/ (id)**

Get information about a Network run

**GET /swaggerui/ (path: filename)**

Function used internally to send static files from the static folder to the browser.

New in version 0.5.

**GET /static/ (path: filename)**

Function used internally to send static files from the static folder to the browser.

New in version 0.5.

# CHAPTER 4

---

## FASTR Developer Module reference

---

### fastr Package

#### fastr Package

FASTR is a top level package which includes all parts required to create networks and edit networks.

##### **class fastr.\_\_init\_\_.Network**

The class representing a Network, this is in fact a reference to `fastr.core.network.Network`.

##### **class fastr.\_\_init\_\_.Node**

The class representing a Node, this is in fact a reference to `fastr.core.node.Node`.

##### **class fastr.\_\_init\_\_.Link**

The class representing a Link, this is in fact a reference to `fastr.core.link.Link`.

##### **class fastr.\_\_init\_\_.SourceNode**

The class representing a data source, this is in fact a reference to `fastr.core.node.SourceNode`.

##### **class fastr.\_\_init\_\_.SinkNode**

The class representing a data sink, this is in fact a reference to `fastr.core.node.SinkNode`.

##### **class fastr.\_\_init\_\_.ConstantNode**

The class representing a constant data source, this is in fact a reference to `fastr.core.node.ConstantNode`.

##### **fastr.\_\_init\_\_.toollist**

A `fastr.core.toolmanager.ToolManager` containing all Tools known to the FASTR environment. The toollist can be accessed in a similar way to a dict. Indexing with a tool id will return the newest version of the Tool. If a specific version of the tool is required a tuple can be used as the index:

```
>>> import fastr
>>> fastr.toollist['testtool']
<Tool: testtool version: 4.2>
>>> fastr.toollist['testtool', '2.0']
<Tool: testtool version: 2.0>
```

##### **fastr.\_\_init\_\_.typelist**

A `fastr.core.datatypemanager.DataTypeManager` containing all Types known to the

FASTR environment. This is usable as a dict where the key is the datatype id and the value is the datatype itself.

```
class fastr.__init__.Network(id_='unnamed_network', version=None)
Bases: fastr.core.serializable.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'
__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`

```
__init__(id_='unnamed_network', version=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.core.network'
```

```
__ne__(other)
```

Tests for non-equality, this is the negated version `__eq__`

```
__repr__()
```

```
__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()
```

```
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*, *nodegroup=None*, *source-group=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
- **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*, *id=None*)

**create\_node** (*tool*, *id*=*None*, *stepid*=*None*, *cores*=*None*, *memory*=*None*, *walltime*=*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
- **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*)

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

**Returns** the newly created sink node

**Return type** *SinkNode*

**create\_source** (*datatype*, *id*=*None*, *stepid*=*None*, *nodegroup*=*None*, *sourcegroup*=*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
- **nodegroup** (*str*) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.
- **sourcegroup** (*str*) – *DEPRECATED!* The nodegroup this SourceNode will be added to

**Returns** the newly created source source\_node

**Return type** *SourceNode*

**draw\_network** (*name*=’network\_layout’, *img\_format*=’svg’, *draw\_dimension*=*False*)

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*, *execution\_plugin*=*None*, *tmpdir*=*None*, *cluster\_queue*=*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

**fullid**

The fullid of the Network

**id**

The id of the Network. This is a read only property.

**is\_valid()****job\_finished** (*job, execution\_interface*)

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

**remove** (*value*)

Remove an item from the Network.

**Parameters** **value** (*Node* or *Link*) – the item to remove

**run\_id**

The run id of the network

**test** (*reference\_data\_dir, source\_data=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

**class** *fastr.\_\_init\_\_.Link* (*source, target, parent=None, id\_=None, collapse=None, expand=None*)  
Bases: *fastr.core.updateable.Updateable, fastr.core.serializable.Serializable*

Class for linking outputs (*BaseOutput*) to inputs (*BaseInput*)

Examples:

```
>>> import fastr
>>> network = fastr.Network()
>>> link1 = network.create_link( n1.outputs['out1'], n2.inputs['in2'] )

link2 = Link()
link2.source = n1.outputs['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

**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`

**\_\_init\_\_(source, target, parent=None, 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, if `None` is given the `fastr.current_network` is assumed to be the parent
- **id** (`str` or `None`) – the id of the link, if no `id_` is given, the id will be in the form of “link\_{:d}”
- **collapse** (`int`, `str`, or tuple of `int/str`) – the dimensions that the link has to collapse on
- **expand** (`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 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.core.link’**

**\_\_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

#### **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

#### **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 links becomes disconnected. This makes sure there is no dangling links.

#### **dimnames**

The dimension names for this Link. The dimension names depend on the connected source output and the collapse/expand.

#### **expand**

Flag indicating that the link will expand the cardinality into a new sample dimension to be created.

#### **fullid**

The full defining ID for the Input

#### **parent**

The Network to which this Link belongs.

#### **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.

#### **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`

**status****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`

```
class fastr.__init__.Node(tool, id_=None, parent=None, cores=None, memory=None, wall-
                           time=None)
Bases:      fastr.core.updateable.Updateable,      fastr.core.serializable.
                           Serializable
```

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)**

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` (`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

**\_\_init\_\_(tool, id\_=None, parent=None, cores=None, memory=None, walltime=None)**

Instantiate a 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
- `cores` (`int`) – number of cores required for executing this Node
- `memory` (`str`) – amount of memory required in the form d+[mMgG] where M is for megabyte and G for gigabyte
- `walltime` (`str`) – amount of time required in second or in the form HOURS:MINUTES:SECOND

**Returns** the newly created Node

**\_\_metaclass\_\_**

alias of ABCMeta

**\_\_module\_\_ = 'fastr.core.node'****\_\_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

**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.

**create\_job** (*sample\_id*, *sample\_index*, *job\_data*, *job\_dependencies*, *\*\*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*)

**dimnames**  
Names of the dimensions in the Node output. These will be reflected in the SampleIdList of this Node.

**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*)

**fullid**  
The full defining ID for the Node

**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

**id**  
The id of the Node

**input\_groups**  
**A list of input groups for this Node. An input group is InputGroup object filled according to the Node**

**listeners**  
All the listeners requesting output of this node, this means the listeners of all Outputs and SubOutputs

**merge\_dimensions**

**name**  
Name of the Tool the Node was based on. In case a Toolless Node was used the class name is given.

**nodegroup**

**outputszie**  
Size of the outputs in this Node

**parent**

The parent network of this node.

**prepare()**

Prepare the node for execution. It will create a SampleIdList of the correct size and prepare the outputs.

**required\_cores**

Number of cores required for the execution of this Node

**required\_memory**

Amount of memory required for the execution of this Node. Follows the format d+[mMgG] so 500M or 4g would be valid ways to specify 500 megabytes or 4 gigabyte of memory.

**required\_time**

Amount of time required for the execution of this Node. Follows the format of a number of second or H:M:S, with H the number of hours, M the number of minutes and S the number of seconds.

**set\_result(job, failed\_annotation)**

Incorporate result of a job into the Node.

**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

**status****tool****update\_input\_groups()**

Update all input groups in this node

**class fastr.\_\_init\_\_.ConstantNode(datatype, data, id\_=None)**

Bases: *fastr.core.node.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)**

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.Network()
>>> source = network.create_source(datatype=fastr.typeList['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.core.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

**data**

The data stored in this constant node

**execute()**

Execute the constant node and create the jobs that need to run

**Returns** list of jobs to run

**Return type** list of *Jobs*

**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.\_\_init\_\_.SourceNode(datatype, id\_=None)**

Bases: *fastr.core.node.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'

**\_\_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** (*Node*) – the other instances to compare to

**Returns** True if equal, False otherwise

**\_\_getstate\_\_()**

Retrieve the state of the SourceNode

**Returns** the state of the object

**Rtype** dict

**\_\_init\_\_(datatype, id\_=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.Network()
>>> source = network.create_source(datatype=fastr.typeList['ITKImageFile'],
    ↵ id_='sourceN')
```

**\_\_module\_\_** = 'fastr.core.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

**create\_job** (*sample\_id*, *sample\_index*, *job\_data*, *job\_dependencies*)

**datatype**

The datatype of the data this source supplies.

**dimnames**

Names of the dimensions in the SourceNode output. These will be reflected in the SampleIdLists.

**execute()**

Execute the source node and create the jobs that need to run

**Returns** list of jobs to run

**Return type** list of *Jobs*

**output**

Shorthand for `self.outputs['output']`

**outputszie**

The size of output of this SourceNode

**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

**sourcegroup**

**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.

## 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.configmanager.Config(*configfiles)
Bases: object
```

Class contain the fastr configuration

```
DEFAULT_FIELDS = {'resourcesdir': (<type 'str'>, '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/resources')}
```

```
__dict__ = dict_proxy({'resourcesdir': <property object>, '__module__': 'fastr.configmanager', 'plugins_path': <property object>})
```

---

```

__init__(*configfiles)
__module__ = 'fastr.configmanager'
__repr__()
__weakref__
    list of weak references to the object (if defined)

debug
examplesdir
execution_plugin
executionscript
get_field(item)
logdir
logtype
mounts
networks_path
pim_host
plugins_path
preferred_types
process_pool_worker_number
protected_modules
read_config(filename)
    Read a configuration and update the configuration object accordingly

        Parameters filename – the configuration file to read

read_config_files = None
    Trace of the config files read by this object

register_fields(fields_spec)
    Register extra fields to the configuration manager.

resourcesdir
schemadir
set_field(item, value)
source_job_limit
systemdir
tools_path
types_path
userdir
warn_develop
web_hostname
web_port
web_secret_key
web_url()
    Construct a fqdn from the web['hostname'] and web['port'] settings. :return: FQDN :rtype: str
x = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.e

```

```
class fastr.configmanager.EmptyDefault(data=None)
Bases: object

__add__(right)
__delitem__(key)
__dict__ = dict_proxy({'extend': <function extend>, '__delitem__': <function __delitem__>, '__module__': 'fastr.c
__getitem__(item)
__iadd__(right)
__init__(data=None)
__module__ = 'fastr.configmanager'
__radd__(other)
__setitem__(key, value)
__weakref__
    list of weak references to the object (if defined)

append(value)
asdict()
aslist()
extend(other)
prepend(value)
update(other)
```

## datatypes Module

The `datatypes` module holds all `DataTypes` generated by `fastr` and all the base classes for these datatypes.

```
class fastr.datatypes.AnalyzeImageFile (value=None, format_=None)
    Bases: fastr.datatypes.URLType

    __abstractmethods__ = frozenset(())
    __module__ = 'fastr.datatypes'

    classmethod content (invalue, outvalue=None)
        description = 'Analyze Image file formate'
        extension = 'hdr'

        filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'
        module = <module 'AnalyzeImageFile' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'>
```

```
class fastr.datatypes.AnyFile(value=None)  
    Bases: fastr.datatypes.TypeGroup
```

Special Datatype in fastr that is a TypeGroup with all known DataTypes as its members.

abstractmethods = frozenset([])

**\_\_module\_\_** = ‘fastr.datatypes’

```
description = 'TypeGroup AnyFile\nAnyFile'
```

Bases: [fastr.datatypes.TypeGroup](#)

```

__abstractmethods__ = frozenset([])
__module__ = 'fastr.datatypes'

description = 'TypeGroup AnyType\nAnyType (AnyType) is a group of consisting of all DataTypes known by fastr'

class fastr.datatypes.BaseDataType (value=None, format_=None)
Bases: fastr.core.baseplugin.BasePlugin

The base class for all datatypes in the fastr type system.

__abstractmethods__ = frozenset(['__init__'])

__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__()

__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)

__repr__()
Returns string representation of the BaseDataType

Returns string represenation
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 =

```

```
extension = None
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2'
fullId = 'fastr://typelist/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>
parsed_value
    The parsed value of object instantiation of this DataType.

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

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.

value
    The value of object instantiation of this DataType.

version = <Version: 1.0>

class fastr.datatypes.Boolean(value=None, format_=None)
Bases: fastr.datatypes.ValueType

Datatype representing a boolean

__abstractmethods__ = frozenset([])
__module__ = 'fastr.datatypes'
__str__()

description = 'A boolean value (True or False)'

filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2'
module = <module 'Boolean' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2/Boolean.py'>
value
    The value of object instantiation of this DataType.

class fastr.datatypes.DataType(value=None, format_=None)
Bases: fastr.datatypes.BaseDataType

This class is the base class for all DataTypes that can hold a value.

__abstractmethods__ = frozenset(['__init__'])
__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

---

**class** *fastr.datatypes.Deferred* (*value=None, format\_=None*)  
Bases: *fastr.datatypes.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*

**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 deferrred url

**parsed\_value**  
The value of object instantiation of this DataType.

**provenance**

**target**  
Target object for this deferred.

**Raises**

- *FastrKeyError* – if the deferred is not available (yet)
- *FastrValueError* – if the value is not a valid deferrred url

**value**

The value of object instantiation of this DataType.

**class** `fastr.datatypes.Directory` (*value=None, format\_=None*)  
Bases: `fastr.datatypes.URLType`

DataType representing a directory.

**\_\_abstractmethods\_\_ = frozenset([])**

**\_\_eq\_\_ (other)**

Directories are equal by default as long as the validity matches.

**Parameters other** (`Directory`) – other to compare against

**Returns** equality flag

**\_\_module\_\_ = 'fastr.datatypes'**

**action (name)**

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

**description = 'A directory on the disk'**

**extension = None**

**filename = '/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info'**

**module = <module 'Directory' from '/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info'**

**class** `fastr.datatypes.EnumType` (*value=None, format\_=None*)

Bases: `fastr.datatypes.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)**

**description = 'EnumType (EnumType) is a enumerate type with options:\n\n\nEnumType can take the value of an integer or string.'**

**options = frozenset([])**

**version = <Version: 1.0>**

**class** `fastr.datatypes.FilePrefix` (*value=None, format\_=None*)

Bases: `fastr.datatypes.URLType`

```

__abstractmethods__ = frozenset([])
__module__ = 'fastr.datatypes'

checksum()
description = 'Prefix for another file'
extension = 'prefix'
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'
module = <module 'FilePrefix' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'>

class fastr.datatypes.Float (value=None, format_=None)
Bases: fastr.datatypes.ValueType

__abstractmethods__ = frozenset([])
__module__ = 'fastr.datatypes'
description = 'A floating point value'
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'
module = <module 'Float' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'>
value
The value of object instantiation of this DataType.

class fastr.datatypes.ITKImageFile (value=None)
Bases: fastr.datatypes.TypeGroup

__abstractmethods__ = frozenset([])
__module__ = 'fastr.datatypes'
description = 'Text file to store point coordinates'
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'
module = <module 'ITKImageFile' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'>

class fastr.datatypes.Int (value=None, format_=None)
Bases: fastr.datatypes.ValueType

__abstractmethods__ = frozenset([])
__module__ = 'fastr.datatypes'
description = 'an integer value'
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'
module = <module 'Int' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'>
value
The value of object instantiation of this DataType.

class fastr.datatypes.JsonFile (value=None, format_=None)
Bases: fastr.datatypes.URLType

__abstractmethods__ = frozenset([])
__module__ = 'fastr.datatypes'
checksum()
description = 'json file'
extension = 'json'
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'
module = <module 'JsonFile' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes.py'>

```

```
class fastr.datatypes.MetaImageFile (value=None, format_=None)
    Bases: fastr.datatypes.URLType

    __abstractmethods__ = frozenset([])
    __eq__ (other)
    __module__ = 'fastr.datatypes'

checksum ()
    Return the checksum of this MetaImageFile

        Returns checksum string

    Return type str

classmethod content (invalue, outvalue=None)

    description = 'Meta Image file format'

    extension = 'mhd'

    filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/metaimagefile.py'
    module = <module 'MetaImageFile' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/metaimagefile.py'>

class fastr.datatypes.NiftiImageFile (value=None)
    Bases: fastr.datatypes.TypeGroup

    __abstractmethods__ = frozenset([])
    __module__ = 'fastr.datatypes'

    description = 'NiftiTypeGroup'

    filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/niftiimagefile.py'
    module = <module 'NiftiImageFile' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/niftiimagefile.py'>

class fastr.datatypes.NiftiImageFileCompressed (value=None, format_=None)
    Bases: fastr.datatypes.URLType

    __abstractmethods__ = frozenset([])
    __module__ = 'fastr.datatypes'

    description = 'Compressed Nifti Image File format'

    extension = 'nii.gz'

    filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/niftiimagefilecompressed.py'
    module = <module 'NiftiImageFileCompressed' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/niftiimagefilecompressed.py'>

class fastr.datatypes.NiftiImageFileUncompressed (value=None, format_=None)
    Bases: fastr.datatypes.URLType

    __abstractmethods__ = frozenset([])
    __module__ = 'fastr.datatypes'

    description = 'Nifti Image File format'

    extension = 'nii'

    filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/niftiimagefileuncompressed.py'
    module = <module 'NiftiImageFileUncompressed' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/niftiimagefileuncompressed.py'>

class fastr.datatypes.Number (value=None)
    Bases: fastr.datatypes.TypeGroup

    __abstractmethods__ = frozenset([])
    __module__ = 'fastr.datatypes'
```

```

description = ‘an numeric value’
filename = ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/ProvNFile.py’
module = <module ‘Number’ from ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/__init__.py’>

class fastr.datatypes.ProvNFile (value=None, format_=None)
Bases: fastr.datatypes.URLType

__abstractmethods__ = frozenset([()])
__module__ = ‘fastr.datatypes’

description = ‘Provenance file in the prov N format’
extension = ‘provn’
filename = ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/ProvNFile.py’
module = <module ‘ProvNFile’ from ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/__init__.py’>

class fastr.datatypes.String (value=None, format_=None)
Bases: fastr.datatypes.ValueType

__abstractmethods__ = frozenset([()])
__module__ = ‘fastr.datatypes’

description = ‘A simple string value’
filename = ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/String.py’
module = <module ‘String’ from ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/__init__.py’>

class fastr.datatypes.TifImageFile (value=None, format_=None)
Bases: fastr.datatypes.URLType

__abstractmethods__ = frozenset([()])
__module__ = ‘fastr.datatypes’

description = ‘Tif Image File format’
extension = ‘tif’
filename = ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/TifImageFile.py’
module = <module ‘TifImageFile’ from ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/__init__.py’>

class fastr.datatypes.TxtFile (value=None, format_=None)
Bases: fastr.datatypes.URLType

__abstractmethods__ = frozenset([()])
__module__ = ‘fastr.datatypes’

description = ‘General text file’
extension = ‘txt’
filename = ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/TxtFile.py’
module = <module ‘TxtFile’ from ‘/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/datatypes/__init__.py’>

class fastr.datatypes.TypeGroup (value=None)
Bases: fastr.datatypes.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\_\_(value=None, format\_=None)**  
Instantiate a TypeGroup. This will 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)**

**members**  
A descriptor that can act like a property for a class.

**class fastr.datatypes.URLType (value=None, format\_=None)**  
Bases: `fastr.datatypes.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`

**\_\_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(inval, 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**

- **inval** – 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

**parsed\_value**  
The parsed value of object instantiation of this DataType.

**valid**  
A boolean flag that indicates whether or not the value assigned to this DataType is valid. This property is generally overwritten by implementation of specific DataTypes.

```
class fastr.datatypes.UnsignedInt (value=None, format_=None)
```

Bases: *fastr.datatypes.ValueType*

**\_\_abstractmethods\_\_** = frozenset([])

**\_\_module\_\_** = ‘fastr.datatypes’

**description** = ‘an unsigned integer value’

**filename** = ‘/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/typedescription/\_UnsignedInt.py’

**module** = <module ‘UnsignedInt’ from ‘/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/typedescription/\_UnsignedInt.py’>

**value**

The value of object instantiation of this DataType.

```
class fastr.datatypes.ValueType (value=None, format_=None)
```

Bases: *fastr.datatypes.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

## exceptions Module

This module contains all Fastr-related Exceptions

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

Bases: *fastr.exceptions.FastrError, exceptions.AttributeError*

AttributeError in the fastr system

**\_\_module\_\_** = ‘fastr.exceptions’

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

Bases: *fastr.exceptions.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: *fastr.exceptions.FastrError*

The description of the cardinality is not valid.

```
__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrCollectorError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError

    Cannot collect the results from a Job because of an error

__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrDataTypeFileNotFoundException (*args, **kwargs)
    Bases: fastr.exceptions.FastrError

    Could not read the datatype file.

__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrDataTypeMismatchError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError

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

__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrDataTypeNotFoundError (*args, **kwargs)
    Bases: fastr.exceptions.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: fastr.exceptions.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: fastr.exceptions.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: exceptions.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'

__str__()
    String representation of the error

    Returns error string

    Return type str

__weakref__
    list of weak references to the object (if defined)

except ()
    Return a excerpt of the Error as a tuple.
```

---

```
exception fastr.exceptions.FastrErrorInSubprocess (*args, **kwargs)
    Bases: fastr.exceptions.FastrExecutionError
        Encountered an error in the subprocess started by the execution script
        __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrExecutableNotFoundError (executable=None, *args,
                                                       **kwargs)
    Bases: fastr.exceptions.FastrExecutionError
        The executable could not be found!
        __init__ (executable=None, *args, **kwargs)
        __module__ = 'fastr.exceptions'
        __str__ ()
            String representation of the error

exception fastr.exceptions.FastrExecutionError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError
        Base class for all fastr execution related errors
        __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrIOError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError, exceptions.IOError
        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: fastr.exceptions.FastrError, exceptions ImportError
        ImportError in the fastr system
        __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrIndexError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError, exceptions IndexError
        IndexError in the fastr system
        __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrIndexNonexistent (*args, **kwargs)
    Bases: fastr.exceptions.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: fastr.exceptions.FastrError, exceptions KeyError
        KeyError in the fastr system
        __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrLookupError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError
        Could not find specified object in the fastr environment.
        __module__ = 'fastr.exceptions'
```

```
exception fastr.exceptions.FastrMountUnknownError (*args, **kwargs)
    Bases: fastr.exceptions.FastrKeyError
    Trying to access an undefined mount
    __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrNetworkMismatchError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError
    Two interacting objects belong to different fastr network.
    __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrNetworkUnknownError (*args, **kwargs)
    Bases: fastr.exceptions.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.FastrNoValidTargetException (*args, **kwargs)
    Bases: fastr.exceptions.FastrKeyError
    Cannot find a valid target for the tool
    __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrNodeAlreadyPreparedError (*args, **kwargs)
    Bases: fastr.exceptions.FastrStateError
    A attempt is made at preparing a Node 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: fastr.exceptions.FastrStateError
    When trying to access execution data of a Node, the Node must be prepare. The Node has not been prepared by the execution, so the data is not available!
    __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrNodeNotValidError (*args, **kwargs)
    Bases: fastr.exceptions.FastrStateError
    A Node is not in a valid state where it should be, typically an invalid Node is passed to the executor causing trouble.
    __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrNotExecutableError (*args, **kwargs)
    Bases: fastr.exceptions.FastrExecutionError
    The command invoked by subprocess is not executable on the system
    __module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrNotImplementedError (*args, **kwargs)
    Bases: fastr.exceptions.FastrError, exceptions.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: fastr.exceptions.FastrError, exceptions.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: `fastr.exceptions.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: `fastr.exceptions.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.FastrOutputValidationException(*args, **kwargs)`

Bases: `fastr.exceptions.FastrExecutionError`

An output of a Job does not pass validation

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: `fastr.exceptions.FastrError`

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

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: `fastr.exceptions.FastrNotImplementedError`

A plugin did not implement a capability that it advertised.

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: `fastr.exceptions.FastrKeyError`

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

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: `fastr.exceptions.FastrStateError`

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

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: `fastr.exceptions.FastrExecutionError`

Could not find 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.FastrSerializationError(message, serializer, original_exception=None)`

Bases: `fastr.exceptions.FastrError`

The serialization encountered a serious problem

**\_\_init\_\_(message, serializer, original\_exception=None)****\_\_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: *fastr.exceptions.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: *fastr.exceptions.FastrSerializationError*

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

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: *fastr.exceptions.FastrKeyError*

The desired serialization method does not exist.

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: *fastr.exceptions.FastrKeyError*

Could not find the Sink data for the desire sink.

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: *fastr.exceptions.FastrError*

The given size cannot be valid.

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: *fastr.exceptions.FastrError*

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

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: *fastr.exceptions.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: *fastr.exceptions.FastrKeyError*

Could not find the Source data for the desire source.

**\_\_module\_\_ = 'fastr.exceptions'**

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

Bases: *fastr.exceptions.FastrError*

An object is in an invalid/unexpected state.

---

```
__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrSubprocessNotFinished(*args, **kwargs)
    Bases: fastr.exceptions.FastrExecutionError

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

__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrToolNotFoundError(*args, **kwargs)
    Bases: fastr.exceptions.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: fastr.exceptions.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: fastr.exceptions.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.FastrTypeError(*args, **kwargs)
    Bases: fastr.exceptions.FastrError, exceptions.TypeError

    TypeError in the fastr system

__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrUnknownURLSchemeError(*args, **kwargs)
    Bases: fastr.exceptions.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: fastr.exceptions.FastrError, exceptions.ValueError

    ValueError in the fastr system

__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrVersionInvalidError(*args, **kwargs)
    Bases: fastr.exceptions.FastrValueError

    The string representation of the version is malformatted.

__module__ = 'fastr.exceptions'

exception fastr.exceptions.FastrVersionMismatchError(*args, **kwargs)
    Bases: fastr.exceptions.FastrValueError

    There is a mismatch between different parts of the Fastr environment and integrity is compromised.
```

## plugins Module

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,
                                         status_callback=None)
Bases: fastr.execution.executionpluginmanager.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, status_callback=None)
__module__ = 'fastr.plugins'
cleanup()
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2
module = <module 'blockingexecution' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/
```

```
class fastr.plugins.CommaSepreatedValueFile
Bases: fastr.core.iopluging.IOPlugin
```

The CommaSepreatedValueFile 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=,
```

```
__abstractmethods__ = frozenset([])

__init__()

__module__ = 'fastr.plugins'

expand_url (url)
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/CrossValidation.py'
module = <module 'commaseparatedvaluefile' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/CSVPlugin.py'>
scheme = 'csv'

class fastr.plugins.CrossValidation
Bases: flowinterface.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/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/CrossValidation.py'
module = <module 'crossvalidation' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/CrossValidation.py'>

class fastr.plugins.DRMAAExecution (finished_callback=None, cancelled_callback=None, status_callback=None)
Bases: fastr.execution.executionpluginmanager.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

SUPPORTS_CANCEL = True

SUPPORTS_DEPENDENCY = True

SUPPORTS_HOLD_RELEASE = True

__abstractmethods__ = frozenset([])

__init__ (finished_callback=None, cancelled_callback=None, status_callback=None)
__module__ = 'fastr.plugins'

cleanup ()
collect_jobs ()
configuration_fields = {'drmaa_queue': (<type 'str'>, 'week', 'The default queue to use for jobs send to the system')}.
```

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/FastrInterface.py'
module = <module 'drmaaexecution' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/drmaaexecution.py'>
send_job(command, arguments, queue=None, walltime=None, job_name=None, memory=None, ncores=None, joinLogFiles=False, outputLog=None, errorLog=None, hold_job=None, hold=False)
submit_jobs()
classmethod test()

class fastr.plugins.FastrInterface(id_, document)
    Bases: fastr.core.interface.Interface

The default Interface for fastr. For the command-line Tools as used by fastr.

__abstractmethods__ = frozenset(())
__dataschemafile__ = 'FastrInterface.schema.json'
__eq__(other)
__getstate__()
Get the state of the FastrInterface object.

    Returns state of interface
    Return type dict
__init__(id_, document)
__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
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]
```

**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*

**expanding**

```
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info/expanding.py'
```

**get\_arguments** (*values*)

Get the argument list for this interface

**Returns** return list of arguments

**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

**inputs**

```
module = <module 'fastrinterface' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info/expanding.py'>
```

**outputs****class** *fastr.plugins.FileSystem*

Bases: *fastr.core.ioplugin.IOPPlugin*

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__()
__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/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/_file.py'
```

```
module = <module 'filesystem' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/_file.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: fastr.core.interface.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)
```

```
__getstate__()
```

Get the state of the FastrInterface object.

**Returns** state of interface

**Return type** dict

```
__init__(id_, document)
```

```
__module__ = 'fastr.plugins'
```

```
__setstate__(state)
```

Set the state of the Interface

```

execute (target, payload)
expanding
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/plugins/linearexecution.py'
flow_plugin_type
    alias of FlowPlugin
flow_plugins = FlowPluginManager [37m|42m[1mLoaded[0m CrossValidation : <FlowPlugin: CrossValidation>
inputs
module = <module 'flowinterface' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/plugins/flowinterface.py'
outputs

class fastr.plugins.LinearExecution (finished_callback=None, cancelled_callback=None, status_callback=None)
Bases: fastr.execution.executionpluginmanager.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 no support that.

__abstractmethods__ = frozenset([])
__init__ (finished_callback=None, cancelled_callback=None, status_callback=None)
__module__ = 'fastr.plugins'
cleanup ()
exec_worker ()

filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/plugins/linearexecution.py'
module = <module 'linearexecution' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/FASTR/plugins/linearexecution.py'

class fastr.plugins.NipypeInterface (id_, nipype_cls=None, document=None)
Bases: fastr.core.interface.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)
__getstate__ ()
__init__ (id_, nipype_cls=None, document=None)
__module__ = 'fastr.plugins'
__setstate__ (state)
```

**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*

**expanding**

**filename** = '/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info/dependency\_links.txt'

**get\_type** (*trait*)

**inputs**

**module** = <module ‘nipypeinterface’ from ‘/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info/dependency\_links.txt’>

**outputs**

**classmethod test ()**

**class** fastr.plugins.**Null**

Bases: *fastr.core.iopluging.IOPluging*

The Null plugin is created 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\_\_ ()**

**\_\_module\_\_** = ‘fastr.plugins’

**filename** = '/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info/dependency\_links.txt'

**module** = <module ‘null’ from ‘/home/docs/checkouts/readthedocs.org/user\_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg-info/dependency\_links.txt’>

**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.**ProcessPoolExecution** (*finished\_callback=None, can\_celled\_callback=None, status\_callback=None, nr\_of\_workers=None*)

Bases: *fastr.execution.executionpluginmanager.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,      status_callback=None,
         nr_of_workers=None)
__module__ = 'fastr.plugins'

cleanup()

configuration_fields = {'process_pool_worker_number': (<type 'int'>, 3, 'Number of workers to use in a process pool'),
                        'filename': '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg',
                        'module': <module 'processpoolexecution' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/RQExecution.py'>,
                        'classmethod test()'}

class fastr.plugins.RQExecution(finished_callback=None,      cancelled_callback=None,      status_callback=None)
Bases: fastr.execution.executionpluginmanager.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, status_callback=None)
__module__ = 'fastr.plugins'

check_finished()

cleanup()

configuration_fields = {'rq_queue': (<type 'str'>, 'default', 'The redis queue to use'), 'rq_host': (<type 'str'>, 'localhost', 'The redis host to use'),
                        'filename': '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg',
                        'module': <module 'rqexecution' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/RQExecution.py'>,
                        'classmethod run_job(job_id, job_command, job_stdout, job_stderr)'}

class fastr.plugins.Reference
Bases: fastr.core.ioplugin.IOPPlugin
```

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__()

__module__ = 'fastr.plugins'

filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg',
module = <module 'reference' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/fastr/plugins/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.VirtualFileSystem**  
Bases: *fastr.core.iopluging.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([])****\_\_init\_\_()**

Instantiate the VFS plugin

**Returns** the VirtualFileSysten plugin**\_\_module\_\_ = ‘fastr.plugins’****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\_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://`

**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'
```

**class fastr.plugins.VirtualFileSystemRegularExpression**

Bases: *fastr.core.ioplugin.IOPPlugin*

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__()  
  
__module__ = 'fastr.plugins'  
  
expand_url(url)  
  
filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.  
module = <module 'virtualfilesystemregularexpression' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr-1.2.  
scheme = 'vfsregex'  
  
class fastr.plugins.VirtualFileSystemValueList  
    Bases: fastr.core.ioplugin.IOPPlugin
```

The `VirtualFileSystemValueList` is an expand-only type of `IOPPlugin`. 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__()

__module__ = 'fastr.plugins'

expand_url(url)

filename = '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/EGG-INFO/dependency_links.txt'
module = <module 'virtualfilesystemvalue' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/EGG-INFO/dependency_links.txt'>
scheme = 'vfslist'

class fastr.plugins.XNATStorage
Bases: fastr.core.ioplugin.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 at 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 <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: project, subject, experiment, scan, or resource
label	false	Indicate the XNAT label should be used as fastr id, options true or false
insecure	false	Change the url scheme to be used to http instead of https
regex	false	Change search to use regex <code>re.match()</code> instead of fnmatch for matching

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__()
__module__ = 'fastr.plugins'
cleanup()
connect(server, path='', insecure=False)
```

```
expand_url(url)
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/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg'

module = <module 'xnatstorage' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr-1.2.2-py2.7.egg/xnatstorage.py'>

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'

server

xnat

fastr.plugins.json
alias of JsonCollector

fastr.plugins.path
alias of PathCollector

fastr.plugins.stdout
alias of StdoutCollector
```

## 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

Returns the version
Rtype str

fastr.version.get_hg_info()
    Read information about the current mercurial branch and revision

Returns tuple containing head revision and branch

fastr.version.get_saved_version()
    Get cached version from file

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**

## Subpackages

**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.

### basemanager Module

This module contains the core class for all managers

**class** `fastr.core.basemanager.BaseManager` (*path=None, recursive=False*)  
Bases: `_abcoll.MutableMapping`

**Baseclass for a Manager, subclasses needs to override the following methods:**

`BaseManager._item_extension`, `BaseManager._load_item()`

**\_item\_extension()**

Abstract property that sets the extension of the files to be loaded by the BaseManager. When scanning for items, only files with this extension will be loaded.

**Returns** desired extension

**Return type** `str`

**Raises** `FastrNotImplementedError` – if property not reimplemented in subclass

**\_load\_item** (*filepath, namespace*)

Abstract method to load an item of the BaseManager. This function is not implemented and needs to be reimplemented by a subclass.

**Parameters**

- **filepath** (*str*) – path of the item to load
- **namespace** (*str*) – the namespace of the item to be loaded

**Returns** the loaded item

**Raises** `FastrNotImplementedError` – if called without being reimplemented by a subclass

**\_\_abstractmethods\_\_ = frozenset(['\_load\_item', '\_item\_extension'])**

**\_\_delitem\_\_** (*key*)

Remove item from the BaseManager

**Parameters** `key` – key of the item to remove

**Returns** None

**Raises** `FastrKeyError` – if the key is not found in the BaseManager

**\_\_getitem\_\_** (*key*)

Retrieve item from BaseManager

**Parameters** `key` – the key of the item to retrieve

**Returns** the value indicated by the key

**Raises `FastrKeyError`** – if the key is not found in the BaseManager

**`__init__(path=None, recursive=False)`**  
The BaseManager constructor

**Parameters**

- **path** (`str or None`) – path to scan for items, or None for no path
- **recursive** (`bool`) – Flag to indicate a recursive search is desired

**Returns** the newly created BaseManager

**Return type** `BaseManager`

**`__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)

**`__len__()`**  
Return the number of items in the BaseManager

**Returns** number of items in the BaseManager

**Return type** `int`

**`__metaclass__`**  
alias of ABCMeta

**`__module__ = 'fastr.core.basemanager'`**

**`__repr__()`**  
Convert the BaseManager to a representation string.

**Returns** Representation string

**Return type** `str`

**`__setitem__(key, value)`**  
Set item in the BaseManager

**Parameters**

- **key** – the key of the item to store
- **value** – the value of the item to store

**Returns** None

**`data`**  
The actual data dict underlying this Manager

**`match_filename(filename)`**  
Check if the filename matches the pattern the manager expects.

**Parameters** `filename` – filename to match

**Returns** flag indicating that the filename matches

**`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.

**reload()**  
Reload entire contents of this manager.

## baseplugin Module

The base class for all Plugins in the fastr system

**class fastr.core.baseplugin.BasePlugin**  
Bases: `object`

Base class for Plugins in the fastr system.

**\_\_abstractmethods\_\_ = frozenset([])**

**\_\_dict\_\_ = dict\_proxy({‘status’: <fastr.utils.classproperty.ClassPropertyDescriptor object>, ‘\_\_module\_\_’: ‘fastr.co**

**\_\_init\_\_()**

The BasePlugin constructor.

**Returns** the created plugin

**Return type** `BasePlugin`

**Raises** `FastrPluginNotLoaded` – if the plugin did not load correctly

**\_\_metaclass\_\_**

alias of `PluginMeta`

**\_\_module\_\_ = ‘fastr.core.baseplugin’**

**\_\_repr\_\_()**

**\_\_str\_\_()**

Create string representation of the plugin.

**Returns** string representation

**Return type** `str`

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**cleanup()**

Perform any cleanup action needed when the plugin use ended. This can be closing files/streams etc.

**configuration\_fields = {}**

**fullid = ‘fastr://plugins/BasePlugin’**

**id = ‘BasePlugin’**

**instantiate = False**

**module = None**

**classmethod register\_configuration()**

Register and test the configuration fields of the plugin

**classmethod set\_code (source\_code)**

Set the filename and source code of the plugin

**Parameters** `source_code (str)` – the source code of the plugin

**classmethod set\_status (status, message, exception=None)**

Update the status of the plugin

**Parameters**

• **status (str)** – the new status

• **message (str)** – message explaining the status change

- **exception** (*str*) – stacktrace of the exception causing the failed load

```
source_code
A descriptor that can act like a property for a class.

status = '\x1b[46mUnInitialized\x1b[0m'
status_message = 'Plugin object created'

classmethod test ()
Test the plugin, default behaviour is just to instantiate the plugin

class fastr.core.baseplugin.Plugin
Bases: fastr.core.baseplugin.BasePlugin

__abstractmethods__ = frozenset(())
__module__ = 'fastr.core.baseplugin'

class fastr.core.baseplugin.PluginMeta
Bases: abc.ABCMeta

__module__ = 'fastr.core.baseplugin'
__repr__()

class fastr.core.baseplugin.PluginState
Bases: enum.Enum

__format__(format_spec)
__module__ = 'fastr.core.baseplugin'
static __new__(value)
__reduce_ex__(proto)
__repr__()
__str__()

failed = '\x1b[37m\x1b[41m\x1b[1mFailed\x1b[0m'
loaded = '\x1b[37m\x1b[42m\x1b[1mLoaded\x1b[0m'
preload = '\x1b[102mPreLoad\x1b[0m'
uninitialized = '\x1b[46mUnInitialized\x1b[0m'
unloaded = '\x1b[46mUnLoaded\x1b[0m'
```

## datatypemanager Module

This module manages datatypes. These datatypes are python classes generated from the XML/JSON datatype files.

```
class fastr.core.datatypemanager.DataTypeManager
Bases: fastr.core.pluginmanager.BasePluginManager
```

The DataTypeManager hold a mapping of all DataTypes in the fast system and can create new DataTypes from files/data structures.

```
__abstractmethods__ = frozenset(())
__init__()
The DataTypeManager constructor will create a new DataTypeManager and populate it with all DataTypes it can find in the paths set in fastr.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.core.datatypemanager'****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`

**Returns** the newly created subclass of `EnumType`

**Raises** `FastrTypeError` – if the options is not an iterable of `str`

**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

**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.

**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 resolve-datatype 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 resolve-datatype 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

**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

## 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.

`__dict__ = dict_proxy({‘__dict__’: <attribute ‘__dict__’ of ‘Dimension’ objects>, ‘__module__’: ‘fastr.core.dimensio`

`__init__` (`name, size`)

The constructor for the dimension.

**Parameters**

- `name` (`str`) – Name of the dimension
- `size` (`int` or `sympy.Symbol`) – Size fo the dimension

```
__module__ = 'fastr.core.dimension'

__weakref__
    list of weak references to the object (if defined)

class fastr.core.dimension.HasDimensions
    Bases: 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__ = dict_proxy({ '__module__': 'fastr.core.dimension', '__metaclass__': <class 'abc.ABCMeta'>, '_abc_negative_introspection': True })

__metaclass__
    alias of ABCMeta

__module__ = 'fastr.core.dimension'

__weakref__
    list of weak references to the object (if defined)
```

#### dimensions

The dimensions has to be implemented by any subclass. It has to provide a tuple of Dimensions.

**Returns** dimensions

**Return type** tuple

#### dimnames

A tuple containing the dimension names of this object. All items of the tuple are of type str.

#### size

A tuple containing the size of this object. All items of the tuple are of type int or sympy.Symbol.

## 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 chances of breaking the network functionality!

```
class fastr.core.inputoutput.AdvancedFlowOutput(node, description)
    Bases: fastr.core.inputoutput.Output

    __abstractmethods__ = frozenset([])

    __module__ = 'fastr.core.inputoutput'

    dimnames
        The dimnames of AdvancedFlowNodes have the output id appended, as the sizes per output can be different.

class fastr.core.inputoutput.BaseInput(node, description)
    Bases: fastr.core.inputoutput.BaseInputOutput

    Base class for all inputs.

    __abstractmethods__ = frozenset(['_update', 'fullid', '__getitem__', 'num_subinput', 'itersubinputs', 'size'])
```

**`__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 fastr.typeplist

**`__lshift__(other)`****`__module__ = 'fastr.core.inputoutput'`****`__rrshift__(other)`****`create_link_from(value)`****`itersubinputs()`**

Iterator over the SubInputs

**Returns** iterator

example:

```
>>> for subinput in input_a.itersubinputs():
    print subinput
```

**`num_subinput`**

The number of SubInputs in this Input

**`class fastr.core.inputoutput.BaseInputOutput(node, description)`**

Bases: `fastr.core.samples.HasSamples`, `fastr.core.updateable.Updateable`, `fastr.core.serializable.Serializable`

Base class for Input and Output classes. It mainly implements the properties to access the data from the underlying ParameterDescription.

**`__abstractmethods__ = frozenset(['_update', 'fullid', '__getitem__', 'size'])`****`__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 fastr.typeplist

**\_\_iter\_\_()**

This function is blocked to avoid support for iteration using a legacy `__getitem__` method.

**Returns** None

**Raises** `FastrNotImplementedError` – always

**\_\_module\_\_ = ‘fastr.core.inputoutput’****\_\_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)**

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.

**datatype**

The datatype of this Input/Output

**description**

The description object of this input/output

**fullid**

The fullid of the Input/Output, the fullid should be unique and makes the object retrievable by the network.

**id**

Id of the Input/Output

**node**

The Node to which this Input/Output belongs

**numel**

The number of elements in this Input/Output

**required**

Flag indicating that the Input/Output is required

**size**

The size of the Input/Output

---

```

class fastr.core.inputoutput.BaseOutput (node, description)
Bases: fastr.core.inputoutput.BaseInputOutput

Base class for all outputs.

__abstractmethods__ = frozenset(['_update', 'fullid', '__getitem__', 'size'])

__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
    • FastrDataTypeNotFoundError – if the DataType requested cannot be
        found in the fastr.typelist

__module__ = 'fastr.core.inputoutput'

automatic
Flag indicating that the Output is generated automatically without being specified on the command
line

class fastr.core.inputoutput.Input (node, description)
Bases: fastr.core.inputoutput.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 (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 correspond-
ing 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, description)**

Instantiate an input.

**Parameters**

- **node** ([Node](#)) – the parent node of this input.
- **description** ([ParameterDescription](#)) – the ParameterDescription of the input.

**Returns** the created Input

**\_\_module\_\_ = 'fastr.core.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()****datatype**

The datatype of this Input

**dimnames**

The list names of the dimensions in this Input. This will be a list of str.

**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(key)**  
Get a requested *SubInput*

**Parameters** **key** (*int*) – the index of the *SubInput* to retrieve

**Returns** requested *SubInput*

**index(value)**  
Find index of a SubInput

**Parameters** **value** (*SubInput*) – the *SubInput* to find the index of

**Returns** key

**Return type** int, str

**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.itersubinputs():
    print subinput
```

**num\_subinput**  
The number of SubInputs in this Input

**prepare(sample\_size=None)**  
This function makes sure the *SampleIdList* has the correct size.

**Parameters** **sample\_size** (*tuple of int*) – the required size of the *SampleIdList*. If no size is given, *self.size* will be used by default.

**remove(value)**  
Remove a SubInput from the SubInputs list based on the connected Link.

**Parameters** **value** (*SubInput*, <fastr.core.inputoutput.SubInput>) – the *SubInput* or *SubLink* to removed from this Input

**set\_subinput(key, value)**  
Set a specified SubInput

**Parameters**

- **key** (`int`) – positive integer for position in `_source` list
- **value** – new `SubInput` to assign to the selected location

**size**

The size of the sample collections that can accessed via this Input.

**source**

The mapping of `SubInputs` that are connected and have more than 0 elements.

**class** `fastr.core.inputoutput.Output(node, description)`

Bases: `fastr.core.inputoutput.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` (`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` (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, 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
- *FastrDataTypeNotFoundError* – if the DataType requested cannot be found in the fastr.typeList

**`__module__ = 'fastr.core.inputoutput'`****`__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**`blocking`**

Flag indicating that this Output will cause blocking in the execution

**`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 cast to int
- *FastrValueError* – if the referenced cardinality value cannot be cast to int

**`static create_output_cardinality(desc)`**

Create a lambda function that returns an integer value of the cardinality.

**param** str **desc** The cardinality description string**return** output cardinality description**rtype** tuple

The description string can be one of the following forms:

- N: N number of values needed.
- as:input\_name: the same number of values as attached to input\_name are needed.
- val:input\_name: the number of values needed is the value of input\_name.
- unknown: the output cardinality cannot be estimated a priori

**datatype**

The datatype of this Output

**dimnames**

The list names of the dimensions in this Output. This will be a list of str.

**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)

**listeners**

The list of [Links](#) connected to this Output.

**ndims**

The number of dimensions in this Output

**preferred\_types**

The list of preferred DataTypes for this Output.

**prepare()**

This function makes sure that a value storage will be created

**resulting\_datatype**

The DataType that will the results of this Output will have.

**samples**

The SampleCollection of the samples in this Output. None if the Node has not yet been executed. Otherwise a SampleCollection.

**size**

The sample size of the Output

**valid**

Check if the output is valid, i.e. has a valid cardinality

**class** `fastr.core.inputoutput.SourceOutput` (*node, description*)

Bases: [fastr.core.inputoutput.Output](#)

Output for a SourceNode, this type of Output determines the cardinality in a different way than a normal Node.

**\_\_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 Node has not been executed

**\_\_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 fastr.typeplist

**\_\_module\_\_ = ‘fastr.core.inputoutput’****\_\_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 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

**linearized**

A linearized version of the sample data, this is lazily cached linearized version of the underlying SampleCollection.

**ndims**

The number of dimensions in this SourceOutput

**size**

The sample size of the SourceOutput

**class fastr.core.inputoutput.SubInput (*input\_*)**

Bases: *fastr.core.inputoutput.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, 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.core.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

**description**

**dimnames**

List of dimension names for this SubInput

**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

**input\_group**

The id of the *InputGroup* this SubInputs parent belongs to.

**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
```

**node**

The Node to which this SubInputs parent belongs

**num\_subinput**

The number of SubInputs in this SubInput, this is always 1.

**remove (value)**

Remove a SubInput from parent Input.

**Parameters** **value** (*SubInput*) – the *SubInput* to removed from this Input

**size**

The sample size of the SubInput

**source**

A list with the source *Link*. The list is to be compatible with *Input*

**source\_output**

The *Output* linked to this SubInput

**class** *fastr.core.inputoutput.SubOutput* (*output, index*)

Bases: *fastr.core.inputoutput.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

**\_\_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.core.inputoutput'`

### `__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 cast to int
- `FastrValueError` – if the referenced cardinality value cannot be cast to int

#### `datatype`

The datatype of this SubOutput

#### `fullid`

The full defining ID for the SubOutput

#### `indexrep`

Simple representation of the index.

#### `listeners`

The list of `Links` connected to this Output.

#### `node`

The Node to which this SubOutput belongs

#### `preferred_types`

The list of preferred DataTypes for this SubOutput.

#### `resulting_datatype`

The DataType that will the results of this SubOutput will have.

#### `samples`

The SampleCollection for this SubOutput

## 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`

Bases: `fastr.core.interface.InputSpec`

`__dict__ = dict_proxy({‘__dict__’: <attribute ‘__dict__’ of ‘InputSpec’ objects>, ‘__module__’: ‘fastr.core.interface’}`

`__module__ = ‘fastr.core.interface’`

`static __new__(id_, cardinality, datatype, required=False, description=‘’, default=None, hidden=False)`

### `fastr.core.interface.InputSpecBase`

alias of `InputSpec`

### `class fastr.core.interface.Interface`

Bases: `fastr.core.baseplugin.Plugin, fastr.core.serializable.Serializable`

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

`__abstractmethods__ = frozenset([‘inputs’, ‘execute’, ‘__setstate__’, ‘expanding’, ‘__getstate__’, ‘outputs’])`

`__getstate__()`

Retrieve the state of the Interface

**Returns** the state of the object

**Rtype** dict

**\_\_metaclass\_\_**  
alias of ABCMeta

**\_\_module\_\_ = ‘fastr.core.interface’**

**\_\_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’: {‘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*

**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.

**inputs**  
OrderedDict of Inputs connected to the Interface. The format should be {input\_id: InputSpec}.

**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.InterfacePluginManager**  
Bases: *fastr.core.pluginmanager.PluginSubManager*

Container holding all the CollectorPlugins

**\_\_abstractmethods\_\_ = frozenset([])**

**\_\_init\_\_()**  
Create the Coll :param path: :param recursive: :return:

**\_\_module\_\_ = ‘fastr.core.interface’**

**class fastr.core.interface.InterfaceResult(result\_data, log\_data, 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\_\_ = dict\_proxy({‘\_\_dict\_\_’: <attribute ‘\_\_dict\_\_’ of ‘InterfaceResult’ objects>, ‘\_\_module\_\_’: ‘fastr.core.interface’})**

**\_\_init\_\_(result\_data, log\_data, 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**  
Bases: *fastr.core.interface.OutputSpec*

**\_\_dict\_\_ = dict\_proxy({‘\_\_dict\_\_’: <attribute ‘\_\_dict\_\_’ of ‘OutputSpec’ objects>, ‘\_\_module\_\_’: ‘fastr.core.interface’})**

**\_\_module\_\_ = ‘fastr.core.interface’**

```
static __new__(id_, cardinality, datatype, automatic=True, required=False, description='', hid-
den=False)
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: fastr.core.baseplugin.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

```
__metaclass__
```

alias of ABCMeta

```
__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.

**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

class fastr.core.ioplugin.IOPluginManager
    Bases: fastr.core.pluginmanager.PluginSubManager

    A mapping containing the IOPlugins known to this system

    __abstractmethods__ = frozenset([])
    __init__()
        Create the IOPluginManager and populate it.

        Returns newly created IOPluginManager
    __iter__()
    keytransform(key)
    module = 'fastr.core.ioplugin'
    cleanup()
        Cleanup all plugins, this closes files, connections and other things that could be left dangling otherwise.

    static create_ioplugin_toolexpand_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

    populate()
        Populate the IOPlugins manager. After the default directory scan, add the vfs IOPlugin and create the
        Tools for the IOPlugins

    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**

**fastr.core.ioplugin.main()**

The main entry point for command line access to the IOPlugin

## 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.core.link.Link** (*source, target, parent=None, id\_=None, collapse=None, expand=None*)  
Bases: *fastr.core.updateable.Updateable, fastr.core.serializable.Serializable*

Class for linking outputs (*BaseOutput*) to inputs (*BaseInput*)

Examples:

```
>>> import fastr
>>> network = fastr.Network()
>>> link1 = network.create_link( n1.outputs['out1'], n2.inputs['in2'] )

link2 = Link()
link2.source = n1.outputs['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

**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`

`__init__(source, target, parent=None, 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, if `None` is given the `fastr.current_network` is assumed to be the parent
- **id** (`str` or `None`) – the id of the link, if no **id**\_ is given, the id will be in the form of “link\_{:d}”
- **collapse** (`int`, `str`, or tuple of `int`/`str`) – the dimensions that the link has to collapse on
- **expand** (`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 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.core.link'``__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

**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

**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 links becomes disconnected. This makes sure there is no dangling links.

**dimnames**

The dimension names for this Link. The dimension names depend on the connected source output and the collapse/expand.

**expand**

Flag indicating that the link will expand the cardinality into a new sample dimension to be created.

**fullid**

The full defining ID for the Input

**parent**

The Network to which this Link belongs.

**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.

**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*

**status****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.core.network.Network** (*id\_=’unnamed\_network’*, *version=None*)  
Bases: *fastr.core.serializable.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’**

**\_\_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

**\_\_init\_\_ (id\_=’unnamed\_network’, version=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.core.network’**

**\_\_ne\_\_ (other)**

Tests for non-equality, this is the negated version \_\_eq\_\_

**\_\_repr\_\_ ()**

**\_\_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()****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, nodegroup=None, source\_group=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

- **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*, *id\_=None*)**create\_node** (*tool*, *id\_=None*, *stepid=None*, *cores=None*, *memory=None*, *walltime=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
- **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*)

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

**Returns** the newly created sink node

**Return type** *SinkNode*

**create\_source** (*datatype*, *id\_=None*, *stepid=None*, *nodegroup=None*, *sourcegroup=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
- **nodegroup** (*str*) – The group the node belongs to, this can be important for FlowNodes and such, as they will have matching dimension names.
- **sourcegroup** (*str*) – *DEPRECATED!* The nodegroup this SourceNode will be added to

**Returns** the newly created source source\_node

**Return type** *SourceNode*

**draw\_network** (*name='network\_layout'*, *img\_format='svg'*, *draw\_dimension=False*)

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*, *execution\_plugin=None*, *tmpdir=None*, *cluster\_queue=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

**fullid**

The fullid of the Network

**id**

The id of the Network. This is a read only property.

**is\_valid()**

**job\_finished** (*job*, *execution\_interface*)

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

**namespace = None**

The namespace this network lives in, this will be set by the NetworkManager on load

**remove** (*value*)

Remove an item from the Network.

**Parameters** **value** (*Node* or *Link*) – the item to remove

**run\_id**

The run id of the network

**test** (*reference\_data\_dir*, *source\_data=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

## networkmanager Module

This module contains the tool manager class

**class** *fastr.core.networkmanager.NetworkManager* (*path*)

Bases: *fastr.core.objectmanager.ObjectManager*

---

```

__abstractmethods__ = frozenset([])
__module__ = 'fastr.core.networkmanager'
get_object_version(obj)
object_class

fastr.core.networkmanager.networklist = NetworkManager add_ints v0.0 : /home/docs/checkouts/readthedocs
The fastr networklist

```

## 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.core.node.AdvancedFlowNode(tool, id_=None, parent=None, cores=None, memory=None, walltime=None)
```

Bases: *fastr.core.node.FlowNode*

```
__abstractmethods__ = frozenset([])
```

```
__module__ = 'fastr.core.node'
```

```
execute()
```

Execute the node and create the jobs that need to run

**Returns** list of jobs to run

**Return type** list of *Jobs*

```
set_result(job, failed_annotation)
```

```
class fastr.core.node.ConstantNode(datatype, data, id_=None)
```

Bases: *fastr.core.node.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)
```

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.Network()
>>> source = network.create_source(datatype=fastr.typeList['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.core.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

**data**

The data stored in this constant node

**execute()**

Execute the constant node and create the jobs that need to run

**Returns** list of jobs to run

**Return type** list of *Jobs*

**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.core.node.DefaultInputGroupCombiner** (*parent\_node*)

Bases: *object*

**\_\_dict\_\_** = dict\_proxy({'**\_\_module\_\_**': 'fastr.core.node', '**merge\_sample\_jobsup**'

**\_\_init\_\_(parent\_node)**

**\_\_iter\_\_()**

**\_\_module\_\_** = 'fastr.core.node'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**dimnames**

**fullid**

The full id of the InputGroupCombiner

**input\_groups**

The input groups under management

**iter\_input\_groups()**

**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)
outputsize
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:

```
>>> list_of_items = combiner.unmerge(item)
>>> item = combiner.merge(list_of_items.values())
```

**Parameters** `item` (*list*) – the item to unmerge

**Returns** items

**Return type** OrderedDict

**update** ()

```
class fastr.core.node.FlowNode (tool, id=None, parent=None, cores=None, memory=None,
                               walltime=None)
Bases: fastr.core.node.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*, *cores=None*, *memory=None*, *walltime=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.core.node'

**blocking**

A FlowNode is (for the moment) always considered blocking.

**Returns** True

**dimnames**

Names of the dimensions in the Node output. These will be reflected in the SampleIdList of this Node.

**outputsize**

Size of the outputs in this Node

**set\_result** (*job*, *failed\_annotation*)

Incorporate result of a job into the FlowNode.

**Parameters** `job` (*Type*) – job of which the result to store

```
class fastr.core.node.InputDict(*args, **kwdss)
Bases: collections.OrderedDict
```

The container containing the Inputs of Node. Implements helper functions for the easy linking syntax.

**\_\_module\_\_** = 'fastr.core.node'

**\_\_setitem\_\_(key, value, dict\_setitem=<slot wrapper '\_\_setitem\_\_' of 'dict' objects>)**

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
- **dict\_setitem** – the setitem function to use for the underlying OrderedDict insert

```
class fastr.core.node.InputGroup(*args, **kwargs)
```

Bases: collections.OrderedDict

A class representing a group of inputs. Input groups allow the

**\_\_abstractmethods\_\_** = frozenset([])

**\_\_delitem\_\_(\*args, \*\*kwargs)**

od.\_\_delitem\_\_(y) <==> del od[y]

---

**Note:** This is a wrapped version of `collections.__delitem__` which triggers an update of the object after being called

**\_\_getitem\_\_(key)**

**\_\_init\_\_(\*args, \*\*kwargs)**

Create a new InputGroup representation

#### Parameters

- **parent** ([Node](#)) – the parent node
- **id** (*str*) – the id of the input group

**Raises [FastrTypeError](#)** – if parent is not a Node

---

**Note:** This is a wrapped version of `fastr.core.node.__init__` which triggers an update of the object after being called

**\_\_metaclass\_\_**

alias of UpdateableMeta

**\_\_module\_\_** = 'fastr.core.node'

**\_\_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.core.node.__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']`**

`clear()` → None. Remove all items from od.

---

**Note:** This is a wrapped version of `collections.clear` which triggers an update of the object after being called

---

**`dimnames`**

The names of the dimensions in this InputGroup

**`empty`**

Bool indicating that this InputGroup is empty (has no data connected)

**classmethod `find_source_index` (`target_size, target_dimnames, source_size, source_dimnames, target_index`)**

**`fullid`**

The full id of the InputGroup

**`iterinputvalues`**

Iterate over the item in this InputGroup

**Returns** iterator yielding SampleItems

**`parent`**

The parent node of this InputGroup

`pop(k[, d])` → v, remove specified key and return the corresponding

**value. If key is not found, d is returned if given, otherwise `KeyError` is raised.**

---

**Note:** This is a wrapped version of `collections.pop` which triggers an update of the object after being called

---

`popitem()` → (k, v), return and remove a (key, value) pair.

Pairs are returned in LIFO order if last is true or FIFO order if false.

---

**Note:** This is a wrapped version of `collections.popitem` which triggers an update of the object after being called

---

**`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.

`setdefault(k[, d])` → od.get(k,d), also set od[k]=d if k not in od

---

**Note:** This is a wrapped version of `collections.setdefault` which triggers an update of the object after being called

---

**size**

The sample size of this InputGroup

**classmethod** **solve\_broadcast** (*target\_size*, *target\_dimnames*, *source\_size*, *source\_dimnames*,  
*target\_index*, *nodegroups=None*)

**update** ([*E*], \*\**F*) → None. Update D from mapping/iterable E and F.

**If E present and has a .keys() method, does:** **for k in E: D[k] = E[k]** If E present and lacks .keys()  
method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items():  
D[k] = v

---

**Note:** This is a wrapped version of `_abcoll.update` which triggers an update of the object after  
being called

---

**class** *fastr.core.node.MacroNode* (*network*, *id=None*, *parent=None*, *cores=None*, *memory=None*, *walltime=None*)

Bases: *fastr.core.node.Node*

MacroNode encapsulates an entire network in a single node.

**\_\_abstractmethods\_\_ = frozenset([])**

**\_\_getstate\_\_()**

Retrieve the state of the MacroNode

**Returns** the state of the object

**Rtype** dict

**\_\_init\_\_** (*network*, *id=None*, *parent=None*, *cores=None*, *memory=None*, *walltime=None*)

**Parameters** **network** (*Network*) – network to create macronode for

**\_\_module\_\_ = 'fastr.core.node'**

**\_\_setstate\_\_** (*state*)

**execute()**

**class** *fastr.core.node.MergingInputGroupCombiner* (*parent\_node*, *merge\_dimension*)

Bases: *fastr.core.node.DefaultInputGroupCombiner*

**\_\_init\_\_** (*parent\_node*, *merge\_dimension*)

**\_\_module\_\_ = 'fastr.core.node'**

**iter\_input\_groups()**

**merge** (*list\_of\_items*)

**unmerge** (*item*)

**update()**

**class** *fastr.core.node.Node* (*tool*, *id=None*, *parent=None*, *cores=None*, *memory=None*, *walltime=None*)

Bases: *fastr.core.updateable.Updateable*, *fastr.core.serializable.Serializable*

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*)

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` (`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

**\_\_init\_\_(tool, id=None, parent=None, cores=None, memory=None, walltime=None)**  
Instantiate a 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
- `cores` (`int`) – number of cores required for executing this Node
- `memory` (`str`) – amount of memory required in the form d+[mMgG] where M is for megabyte and G for gigabyte
- `walltime` (`str`) – amount of time required in second or in the form HOURS:MINUTES:SECOND

**Returns** the newly created Node

**\_\_metaclass\_\_**  
alias of ABCMeta

**\_\_module\_\_ = ‘fastr.core.node’**

**\_\_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

**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.

**create\_job(sample\_id, sample\_index, job\_data, job\_dependencies, \*\*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)`

**dimnames**  
Names of the dimensions in the Node output. These will be reflected in the SampleIdList of this Node.

**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)**

**fullid**  
The full defining ID for the Node

**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

**id**  
The id of the Node

**id\_=None**  
The Node id is a unique string identifying the Node

**input\_groups**  
A list of input groups for this Node. An input group is `InputGroup` object filled according to the Node

**inputs=None**  
A list of inputs of this Node

**listeners**  
All the listeners requesting output of this node, this means the listeners of all Outputs and SubOutputs

**merge\_dimensions**

**name**  
Name of the Tool the Node was based on. In case a Toolless Node was used the class name is given.

**nodegroup**

**outputs=None**  
A list of outputs of this Node

**outputsize**  
Size of the outputs in this Node

**parent**  
The parent is the Network this Node is part of

**prepare()**  
Prepare the node for execution. It will create a SampleIdList of the correct size and prepare the outputs.

**required\_cores**  
Number of cores required for the execution of this Node

**required\_memory**  
Amount of memory required for the execution of this Node. Follows the format d+[mMgG] so 500M or 4g would be valid ways to specify 500 megabytes or 4 gigabyte of memory.

**required\_time**

Amount of time required for the execution of this Node. Follows the format of a number of second or H:M:S, with H the number of hours, M the number of minutes and S the number of seconds.

**set\_result (job, failed\_annotation)**

Incorporate result of a job into the Node.

**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

**status****tool****update\_input\_groups ()**

Update all input groups in this node

**class fastr.core.node.OutputDict (\*args, \*\*kwds)**

Bases: `collections.OrderedDict`

The container containing the Inputs of Node. Only checks if the inserted values are actually outputs.

**\_\_module\_\_ = 'fastr.core.node'****\_\_setitem\_\_ (key, value, dict\_setitem=<slot wrapper '\_\_setitem\_\_' of 'dict' objects>)**

Set an output.

**Parameters**

- **key** (*str*) – the of the item to set
- **value** (*BaseOutput*) – the output to set
- **dict\_setitem** – the setitem function to use for the underlying OrderedDict insert

**class fastr.core.node.SinkNode (datatype, id\_=None)**

Bases: `fastr.core.node.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\_\_ ()****\_\_init\_\_ (datatype, id\_=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.Network()
>>> sink = network.create_sink(datatype=fastr.typeList['ITKImageFile'], id_= 'SinkN')
```

**\_\_module\_\_ = 'fastr.core.node'****\_\_setstate\_\_ (state)**

**create\_job** (*sample\_id*, *sample\_index*, *job\_data*, *job\_dependencies*)

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`

**datatype**

The datatype of the data this sink can store.

**execute ()**

Execute the sink node and create the jobs that need to run

**Returns** list of jobs to run

**Return type** list of `Jobs`

**input**

The default input of the sink Node

**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 <code>uuid.uuid1()</code> )

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.{extension}` to be generated. In this particular case `name{ext}` can help as it will create either `name.ext` or `name.{ext}`.

---

**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** A set of annotations, None if no errors else containing a tuple describing the errors

**class** `fastr.core.node.SourceNode (datatype, id_=None)`

Bases: `fastr.core.node.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'`

`__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 (Node)` – the other instances to compare to

**Returns** True if equal, False otherwise

`__getstate__ ()`

Retrieve the state of the SourceNode

**Returns** the state of the object

**Rtype** dict

`__init__ (datatype, id_=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.Network()
>>> source = network.create_source(datatype=fastr.typeлист['ITKImageFile'],
    ↵ id_='sourceN')
```

`__module__ = 'fastr.core.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

`create_job (sample_id, sample_index, job_data, job_dependencies)`

**datatype**

The datatype of the data this source supplies.

**dimnames**

Names of the dimensions in the SourceNode output. These will be reflected in the SampleIdLists.

`execute ()`

Execute the source node and create the jobs that need to run

**Returns** list of jobs to run

**Return type** list of `Jobs`

**output**

Shorthand for `self.outputs ['output']`

**outputszie**

The size of output of this SourceNode

**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

**sourcegroup****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.

## objectmanager Module

This module contains the object manager class

**class fastr.core.objectmanager.ObjectManager (path)**

Bases: *fastr.core.basemanager.BaseManager*

Class for managing all the objects loaded in the fastr system

**\_\_abstractmethods\_\_ = frozenset(['object\_class', 'get\_object\_version'])****\_\_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

**\_\_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.core.objectmanager'****get\_object\_version (obj)**

Get the version of a given object

**Parameters** **object** – the object to use

**Returns** the version of the object

**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.core.pluginmanager.*BasePluginManager* (path=None, recursive=False)**

Bases: *fastr.core.basemanager.BaseManager*

Baseclass for PluginManagers, need to override the self.\_plugin\_class

**\_\_abstractmethods\_\_ = frozenset(['plugin\_class'])****\_\_getitem\_\_ (key)**

Retrieve item from BaseManager

**Parameters** **key** – the key of the item to retrieve

**Returns** the value indicated by the key

**Raises** *FastrKeyError* – if the key is not found in the BaseManager

**\_\_init\_\_ (path=None, recursive=False)**

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

**Returns** newly created plugin manager

**Raises** *FastrTypeError* – if self.\_plugin\_class is set to a class not subclassing BasePlugin

**\_\_module\_\_ = 'fastr.core.pluginmanager'****load\_plugin (plugin\_key)****plugin\_class**

The class from which the plugins must be subclassed

**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.

**class fastr.core.pluginmanager.*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.core.pluginmanager’

**\_\_repr\_\_** ()

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**class** *fastr.core.pluginmanager.PluginManager* (*path=None*)

Bases: *fastr.core.pluginmanager.BasePluginManager*

**\_\_abstractmethods\_\_** = frozenset([])

**\_\_init\_\_** (*path=None*)

**\_\_module\_\_** = ‘fastr.core.pluginmanager’

**\_\_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

**plugin\_class**

The plugin manager contains any Plugin subclass

**class** *fastr.core.pluginmanager.PluginSubManager* (*parent, plugin\_class*)

Bases: *fastr.core.pluginmanager.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 managers for only certain types of plugins (e.g. IOPlugins) without loading them multiple times.

**\_\_abstractmethods\_\_** = frozenset([])

**\_\_init\_\_** (*parent, plugin\_class*)

**\_\_module\_\_** = ‘fastr.core.pluginmanager’

**data**

**plugin\_class**

PluginSubManagers only expose the plugins of a certain class

**class** *fastr.core.pluginmanager.PluginsView* (*parent, plugin\_class*)

Bases: *\_abcoll.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*)

**\_\_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.core.pluginmanager'
__setitem__(key, value)
filter_plugin(plugin)

class fastr.core.pluginmanager.plugin_option_type(filename, name, namespace, id)
Bases: tuple

__dict__ = dict_proxy({'__module__': 'fastr.core.pluginmanager', '_make': <classmethod object>, '_replace': <function replace at 0x7f3e330d1a0>})

__getnewargs__()
    Return self as a plain tuple. Used by copy and pickle.

__getstate__()
    Exclude the OrderedDict from pickling

__module__ = 'fastr.core.pluginmanager'

static __new__(_cls, filename, name, namespace, id)
    Create new instance of plugin_option_type(filename, name, namespace, id)

__repr__()
    Return a nicely formatted representation string

__slots__ = ()

filename
    Alias for field number 0

id
    Alias for field number 3

name
    Alias for field number 1

namespace
    Alias for field number 2

```

## 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__ = dict_proxy({'__module__': 'fastr.core.provenance', 'init_provenance': <function init_provenance>, 'collect_input_argument_provenance': <function collect_input_argument_provenance>, 'activity': <function activity>, 'agent': <function agent>, 'collect_provenance': <function collect_provenance>, 'entity': <function entity>, 'data_uri': <function data_uri>, 'weakref': <listproxy object>})

__init__(host=None)
    Initialize the Provenance object.

__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)
    Record an activity (job, timestamp, attributes) in the provenance.

agent(identifier, other_attributes=None)
    Record an agent (job, identifier, attributes) in the provenance.

collect_input_argument_provenance(input_argument)
    Collect the provenance for this input argument.

collect_provenance(job, advanced_flow=False)
    Collect the provenance for this job.

static data_uri(value, job)
    Record a data URI (job, value) in the provenance.

entity(identifier, other_attributes=None)
    Record an entity (job, identifier, attributes) in the provenance.

```

**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)**

## samples Module

This package holds the classes for working with samples.

**class fastr.core.samples.HasSamples**Bases: *object*

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\_\_', 'size'])****\_\_contains\_\_ (item)****\_\_dict\_\_ = dict\_proxy({'\_abc\_cache': <\_weakrefset.WeakSet object>, '\_\_module\_\_': 'fastr.core.samples', '\_\_metaclass\_\_': None})****\_\_getitem\_\_ (item)****\_\_iter\_\_ ()****\_\_metaclass\_\_**

alias of ABCMeta

**\_\_module\_\_ = 'fastr.core.samples'****\_\_weakref\_\_**

list of weak references to the object (if defined)

**ids ()****indexes ()****items ()****iteritems ()****size****class fastr.core.samples.SampleBaseId**Bases: *tuple*

This class represents a sample id. A sample id is a multi-dimensional id that has a simple, consistent string representation.

**\_\_add\_\_ (other)**

Add another SampleId, this allows to add parts to the SampleId in a convenient way.

**\_\_dict\_\_ = dict\_proxy({'\_\_module\_\_': 'fastr.core.samples', '\_\_new\_\_': <staticmethod object>, '\_\_str\_\_': <function str at 0x10c1f10>})****\_\_module\_\_ = 'fastr.core.samples'****static \_\_new\_\_ (\*args)**

Create a new SampleId

**Parameters args** (iterator/iterable of element type or element type) – the strings to make sample id for

**\_\_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.

**\_\_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

---

**class** fastr.core.samples.SampleCollection (*dimnames, parent*)  
Bases: abcoll.MutableMapping

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

**\_\_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)**  
Createa a new SampleCollection

**\_\_iter\_\_()**

Iterate over the indices

**\_\_len\_\_()**

Get the number of samples in the SampleCollections.

**\_\_module\_\_ = 'fastr.core.samples'****\_\_repr\_\_()****\_\_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

**dimnames**

The dimnames of the SampleCollection

**fullid**

The full defining ID for the SampleIdList

**ndims**

The number of dimensions in this SampleCollection

**parent**

The parent object holding the SampleCollection

**size**

The size of the SampleCollection. The size is the largest index in every dimension. For a 2D SampleCollection with 2 entries (10, 2) and (6, 6) the size would be (10, 6). As that is the rectangular grid that contains all data points.

**class fastr.core.samples.SampleId**

Bases: `fastr.core.samples.SampleBaseId`

SampleId is an identifier for data using human readable strings

**\_\_module\_\_ = 'fastr.core.samples'****class fastr.core.samples.SampleIndex**

Bases: `fastr.core.samples.SampleBaseId`

SampleId is an identifier for data using the location in the N-d data structure.

**\_\_module\_\_ = 'fastr.core.samples'****\_\_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

**expand**(size)

Function expanding a slice SampleIndex into a list of non-slice SampleIndex objects

**Parameters** size – the size of the collection to slice

**isslice**

Flag indicating that the SampleIndex is a slice (as opposed to a simple single index).

**class** fastr.core.samples.SampleItem

Bases: fastr.core.samples.SampleItemBase

**\_\_module\_\_** = ‘fastr.core.samples’

**static \_\_new\_\_**(index, id\_, data, jobs=None, failed\_annotations=None)

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 contructed like follows: (job\_id, reason).

**class** fastr.core.samples.SampleItemBase

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\_\_** = dict\_proxy({'index': <property object>, ‘\_\_module\_\_’: ‘fastr.core.samples’, ‘dimensionality’: <property object>})

**\_\_getnewargs\_\_()**

Get new args gives the arguments to use to re-create this object, This is used for serialization.

**\_\_module\_\_** = ‘fastr.core.samples’

**static \_\_new\_\_**(index, id\_, data, jobs=None, failed\_annotations=None)

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 contructed like follows: (job\_id, reason).

**\_\_repr\_\_()**  
Get a string representation for the SampleItem

**Returns** the string representation

**Return type** str

**cardinality**  
The cardinality of this Sample

**static combine(\*args)**  
Combine a number of SampleItems into a new one.

**Parameters** \*args – 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
```

**data**  
The data SampleValue of the SampleItem

**Returns** The value of this SampleItem

**Return type** SampleValue

**dimensionality**  
The dimensionality of this Sample

**failed\_annotations**

**id**  
The sample id of the SampleItem

**Returns** The id of this SampleItem

**Return type** SampleId

**index**  
The index of the SampleItem

**Returns** The index of this SampleItem

**Return type** SampleIndex

**jobs**  
The set of the jobs on which this SampleItem depends

**Returns** The jobs that generated the data for this SampleItem

**Return type** set

**class fastr.core.samples.SamplePayload**  
Bases: *fastr.core.samples.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__(index, id_, data, jobs=None, failed_annotations=None)
    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 contructed like follows: (job\_id, reason).

```
class fastr.core.samples.SampleValue(*args, **kwargs)
Bases: _abcoll.MutableMapping

A collection containing the content of a sample

__abstractmethods__ = frozenset([])

__add__(other)
__delitem__(key)
__getitem__(item)
__getstate__()
__init__(*args, **kwargs)
__iter__()
__len__()
__module__ = 'fastr.core.samples'
__radd__(other)
__repr__()
__setitem__(key, value)
__setstate__(state)
cast(datatype)
is_mapping
is_sequence
itelements()
mapping_part()
sequence_part()
```

## Serializable Module

This package contains the base class and meta class for all serializable objects in the Fastr system.

```
class fastr.core.serializable.PassThroughSerializer
Bases: object

__dict__ = dict_proxy({'__module__': 'fastr.core.serializable', 'dumps': <staticmethod object>, '__dict__': <attribute>})
```

```
__module__ = 'fastr.core.serializable'

__weakref__
    list of weak references to the object (if defined)

static dumps (data)
static loads (data)

class fastr.core.serializable.Serializable
Bases: object

Superclass for all classes that can be serialized.

SERIALIZERS = {'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr/core/serializable.py': dict_proxy({'load': <classmethod object>, '__module__': 'fastr.core.serializable', 'SERIALIZERS': {'/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr/core/serializable.py': dict_proxy({'load': <classmethod object>, '__module__': 'fastr.core.serializable', 'getstate': ()}}}, '__module__': 'fastr.core.serializable'}
```

**\_\_weakref\_\_**  
list of weak references to the object (if defined)

**static dumps (data)**  
Dump the object to a string

**static loads (data)**  
Load the object from a string

**classmethod createobj (state, \_=None)**  
Create object function for generic objects

**Parameters**

- **cls** – The class to create
- **state** – The state to use to create the Link
- **network** – the parent Network

**Returns** newly created Link

**dump (file\_handle, method='json', \*\*kwargs)**  
Dump the object to a file like object.

**Parameters**

- **file\_handle** – file descriptor to write the data to
- **method (str)** – method of final serialization to use (e.g. json, xml, pickle)
- **kwargs** – extra arguments passed to the final serializer

**dumpf (path, method=None, \*\*kwargs)**  
Dump the object to a file

**Parameters**

- **path** – path where to write the file
- **method (str)** – method of final serialization to use (e.g. json, xml, pickle)
- **kwargs** – extra arguments passed to the final serializer

---

**Note:** The dumpf function can determine the method based on the desired output filename. Also, if the filename ends with .gz it will continue search for another extension (so .json.gz could be found) and will then compress the result with gzip.

---

```
dumpfuncs = {'xml': <module 'fastr.utils.xmltodict' from '/home/docs/checkouts/readthedocs.org/user_builds/fastr/envs/1.2.2/lib/python2.7/site-packages/fastr/utils/xmltodict.py'>}
```

**dumps (method='json', \*\*kwargs)**  
Dump the object to a string

**Parameters**

- **method (str)** – method of final serialization to use (e.g. json, xml, pickle)

- **kwargs** – extra arguments passed to the final serializer

**Returns** serialization string

**Return type** str

**classmethod** **get\_serializer** (*filename=None*)

**classmethod** **load** (*file\_handle, method=None, network=None, \*\*kwargs*)

Load the object from a file-like object

#### Parameters

- **cls** – class of the object
- **file\_handle** – file descriptor to write the data to
- **method** (*str*) – method of final serialization to use (e.g. json, xml, pickle)
- **network** – network in which to place the loaded object
- **kwargs** – extra arguments passed to the final serializer

**Returns** newly created object

**Warning:** Unlike the loadf functions, this function does not automatically detect gzip compression. You read a gzip using the gzip.open method, but not but simply opening a stream and hoping this function will function.

**classmethod** **loadf** (*path, method=None, network=None, \*\*kwargs*)

Load the object from a file

#### Parameters

- **cls** – class of the object
- **path** – path where to write the file
- **method** (*str*) – method of final serialization to use (e.g. json, xml, pickle)
- **network** – network in which to place the loaded object
- **kwargs** – extra arguments passed to the final serializer

**Returns** newly created object

**Note:** The loadf function can determine the method of loading based on the filename. Also it can automatically determine whether a file is gzipped.

**classmethod** **loads** (*string, method=None, network=None, \*\*kwargs*)

Load the object from a string

#### Parameters

- **cls** – class of the object
- **string** (*str*) – the string containing the serialized data
- **method** (*str*) – method of final serialization to use (e.g. json, xml, pickle)
- **network** – network in which to place the loaded object
- **kwargs** – extra arguments passed to the final serializer

**Returns** newly created object

## target Module

The module containing the classes describing the targets.

**class** `fastr.core.target.DockerTarget` (*binary, docker\_image*)

Bases: `fastr.core.target.Target`

A tool target that is located in a Docker images. Can be run using docker-py.

`__abstractmethods__ = frozenset([])`

`__enter__()`

`__exit__(exc_type, exc_value, traceback)`

`__init__(binary, docker_image)`

Define a new docker target.

Parameters `docker_image (str)` – Docker image to use

`__module__ = 'fastr.core.target'`

`container`

`docker_api = None`

Docker api to use for docker target

`monitor_docker(container, resources)`

Monitor a process and profile the cpu, memory and io use. Register the resource use every \_MONITOR\_INTERVAL seconds.

Parameters

- `process (subprocess.Popen)` – process to monitor

- `resources` – list to append measurements to

`run_command(command)`

**class** `fastr.core.target.LocalBinaryTarget` (*binary, paths=None, environment\_variables=None, initscripts=None, modules=None, interpreter=None, \*\*kwargs*)

Bases: `fastr.core.target.Target`

A tool target that is a local binary on the system. Can be found using environmentmodules or vfs-path on the executing machine

`DYNAMIC_LIBRARY_PATH_DICT = {'windows': 'PATH', 'darwin': 'DYLD_LIBRARY_PATH', 'linux': 'LD_LIBRARY_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.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 (subprocess.Popen) – process to monitor
        • resources – list to append measurements to

run_command (command)

class fastr.core.target.ProcessUsageCollection
    Bases: _abcoll.Sequence

    __abstractmethods__ = frozenset([])
    __getitem__ (item)
    __init__ ()
    __len__ ()
    __module__ = ‘fastr.core.target’
    aggregate (number_of_points)
    append (value)
    usage_type
        alias of SystemUsageInfo

class fastr.core.target.SystemUsageInfo (timestamp, cpu_percent, vmem, rmem, read_bytes,
                                         write_bytes)
    Bases: tuple

    __dict__ = dict_proxy({‘__module__’: ‘fastr.core.target’, ‘make’: <classmethod object>, ‘timestamp’: <property object>})
    __getnewargs__ ()
        Return self as a plain tuple. Used by copy and pickle.
    __getstate__ ()
        Exclude the OrderedDict from pickling
    __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__ = ()

cpu_percent
    Alias for field number 1

read_bytes
    Alias for field number 4

rmem
    Alias for field number 3

timestamp
    Alias for field number 0

vmem
    Alias for field number 2

write_bytes
    Alias for field number 5

```

**class** fastr.core.target.Target  
Bases: object

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'])  
...     target.run_command(['sleep', '10'])  
...     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'])

**\_\_dict\_\_** = dict\_proxy({‘\_\_module\_\_’: ‘fastr.core.target’, ‘\_\_abstractmethods\_\_’: frozenset([‘run\_command’]), ‘\_\_weakref\_\_’: {}})

**\_\_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

**\_\_metaclass\_\_**

alias of ABCMeta

**\_\_module\_\_** = ‘fastr.core.target’

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**run\_command** (command)

## 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.
- Output ParameterDescription – A class containing the description of an output parameter.

**class** fastr.core.tool.Tool (doc=None)  
Bases: fastr.core.serializable.Serializable

The class encapsulating a tool.

**\_\_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

```
__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 = None
    List of authors of the tool. These people wrapped the executable but are not responsible for executable
    itself.

cite = None
    This holds the citation you should use when publishing something based on this Tool

command = None
    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 dic-
    tionary 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 descrip-
    tion of the command command['authors'] lists the original authors of the command

command_version

description = None
    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

fullid
    The full id of this tool

hash

help = None
    Man page for the Tool. Here usage and examples can be described in detail

inputs

interface_class = None
    Create the Interface based on the class specified in the tool file

name = None
    Name of the tool, this should be a descriptive, human readable name.

namespace = None
    The namespace this tools lives in, this will be set by the ToolManager on load
```

**node\_class = None**

Class for of the Node to use

**ns\_id**

The namespace and id of the Tool

**outputs****path**

The path of the directory in which the tool definition file was located.

**references = None**

A list of documents and in depth reading about the methods used in this tool

**regex = None**

Identifier for the tool

**requirements = None**

Requirements for this Tool

**Warning:** Not yet implemented

**tags = None**

List of tags for this tool

**target**

The OS and arch matched target definition.

**test()**

Run the tests for this tool

**test\_spec**

alias of TestSpecification

**tests = None**

Test for this tool. A test should be a collection of inputs, parameters and outputs to verify the proper functioning of the Tool.

The format of the tests is a list of namedtuples, that have 3 fields: - input: a dict of the input data  
- command: a list given the expected command-line arguments - output: a dict of the output data to validate

**Warning:** Not yet implemented

**url = None**

URL to website where this tool can be downloaded from

**version = None**

Version of the tool, not of the underlying software

## toolmanager Module

This module contains the tool manager class

```
class fastr.core.toolmanager.ToolManager(path)
    Bases: fastr.core.objectmanager.ObjectManager

    __abstractmethods__ = frozenset(())
    __module__ = 'fastr.core.toolmanager'
    get_object_version(obj)
```

```
object_class
populate()
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.

fastr.core.toolmanager.toollist = ToolManager fastr.Sink v1.0 : /home/docs/checkouts/readthedocs.org/user_bu
    The fastr toollist
```

## updateable Module

This module contains both the Updateable class and UpdateableMeta meta-class for objects which support updates within the fastr system

```
class fastr.core.updateable.Updateable
    Bases: object
```

Super class for all classes that can be updated and have a status. These objects can be valid/invalid and ready/not-ready depending on their state. These states are set by the function update. This allows for interactively checking the network.

```
__abstractmethods__ = frozenset(['_update'])

__dict__ = dict_proxy({'__module__': 'fastr.core.updateable', '_update': <function _update>, '__metaclass__': <cl

__getstate__()
    Retrieve the state of the object, make sure the status is not part of the description as it will not be valid
    after re-creating the object.

        Returns the state of the object
        Rtype dict

__init__()
    Constructor, creates the status field

        Returns newly created object

__metaclass__
    alias of UpdateableMeta

__module__ = 'fastr.core.updateable'

__setstate__ (state)
    Set the state of the object by the given state. This adds a clean status field, making sure it is not
    unintended, outdated information from before serialization.
```

**Parameters** **state** (*dict*) – The state to populate the object with

```
__updatefunc__ (key=None, forward=True, backward=False)
    Default function for updating, it can be called without key to have a new update started with a new
    key.
```

**Parameters**

- **key** (*int*) – a key for this update, should be different than the last update key
- **forward** (*bool*) – flag indicating to update forward in the network
- **backward** (*bool*) – flag indicating to update backward in the network

```
__updateinprogress__ = <thread.lock object>
    Lock to avoid multiple updates happening at the same time
```

**\_\_updatetriggers\_\_ = []**

Which methods need to be wrapped to trigger an update. Override this value to have the functions automatically wrapped. E.g. `__update_triggers__ = ['append', 'insert', '__setitem__']` to have these functions wrapped.

**\_\_updating\_\_ = True**

Flag to indicate that this object is allowed to update

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**messages**

The messages of the last update

**ready**

Flag indicating that the object is ready

**update (key=None, forward=True, backward=False)**

Default function for updating, it can be called without key to have a new update started with a new key.

**Parameters**

- **key** (`int`) – a key for this update, should be different than the last update key
- **forward** (`bool`) – flag indicating to update forward in the network
- **backward** (`bool`) – flag indicating to update backward in the network

**valid**

Flag indicating that the object is valid

**class fastr.core.updateable.UpdateableMeta**

Bases: `abc.ABCMeta`

A metaclass for objects which are updateable and need some methods/properties to trigger an update.

**\_\_module\_\_ = 'fastr.core.updateable'****static \_\_new\_\_ (mcs, name, parents, dct)****classmethod calcmro (mcs, bases)**

Calculate the Method Resolution Order of bases using the C3 algorithm.

Suppose you intended creating a class K with the given base classes. This function returns the MRO which K would have, *excluding* K itself (since it doesn't yet exist), as if you had actually created the class.

Another way of looking at this, if you pass a single class K, this will return the linearization of K (the MRO of K, *including* itself).

**Parameters** `bases` – the list of bases for which create the MRO

**Returns** the list representing the entire MRO, except the (non-existing) class itself

**Note:** Taken from <http://code.activestate.com/recipes/577748-calculate-the-mro-of-a-class/>

Created by Steven D'Aprano and licensed under the MIT license

**classmethod find\_member (mcs, name, parents, dct)**

Find a member of the class in the same way as Python would if it had a given dict and set of bases

**Parameters**

- **mcs** – metaclass at work
- **name** – name of the class to be created
- **parents** – list of the bases for the new class
- **dct** – the dict of the class being created

**Returns** the firstly resolved member or None if nothing found

**static updatetrigger (fnc)**

Function decorator to make a function trigger an update after being called. This is a way to easily have function trigger an update after setting a value without writing tons of wrapper functions. The function keeps the original docstring and appends a note to it.

## version Module

Module containing the class that represent versions

**class fastr.core.version.Version**

Bases: tuple

Class representing a software version definition. Allows for sorting and extraction of parts.

```
__dict__ = dict_proxy({'status': <property object>, '__module__': 'fastr.core.version', '__new__': <staticmethod object>}
__module__ = 'fastr.core.version'
```

**static \_\_new\_\_ (\*version)**

Class containing a version

Can be constructed by:

```
Version( 'major.$minor.$extra[0].$extra[1]$seperator$status$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

```
__str__()
    Return a string representation of the version

build
    the build number, this is following the status (e.g. for 3.2-beta4, this would be 4)

date_version_matcher = <_sre.SRE_Pattern object>

extra
    extra version extension as a list

extra_string
    extra version extension as a string

major
    major version

minor
    minor version

status
    the status of the version (a, alpha, b, beta, rc or r)

suffix
    the remainder of the version which was not formatted in a known way

version_matcher = <_sre.SRE_Pattern object at 0x27d92c0>
```

## **vfs** Module

This module contains the virtual file system code. This is both an internally used object as well as an IOPlugin.

### **Subpackages**

#### **test** Package

##### **test** Package

###### **testdatatypemanager** Module

###### **testdimension** Module

###### **testnetwork** Module

###### **testnode** Module

###### **testsamples** Module

###### **testtool** Module

###### **testversion** Module

###### **testvfs** 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

```
>>> import fastrdata.url import get_path_from_url
>>> get_path_from_url('vfs://data/temp/blaat1.png')
'/media/data/temp/blaat1.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 scheme 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)

## examples Package

### examples Package

#### generate\_reference Module

`fastr.examples.__generate_reference__.create_reference(example_path)`

`fastr.examples.__generate_reference__.generate_all_references()`

#### add\_ints Module

`fastr.examples.add_ints.create_network()`

`fastr.examples.add_ints.main()`

`fastr.examples.add_ints.sink_data(network)`

`fastr.examples.add_ints.source_data(network)`

**auto\_prefix Module**

```
fastr.examples.auto_prefix.create_network()  
fastr.examples.auto_prefix.main()
```

**collapse Module**

```
fastr.examples.collapse.create_network()  
fastr.examples.collapse.main()  
fastr.examples.collapse.sink_data(network)  
fastr.examples.collapse.source_data(network)
```

**collapse\_expand Module**

```
fastr.examples.collapse_expand.create_network()  
fastr.examples.collapse_expand.main()  
fastr.examples.collapse_expand.sink_data(network)  
fastr.examples.collapse_expand.source_data(network)
```

**cross\_validation Module**

```
fastr.examples.cross_validation.create_network()  
fastr.examples.cross_validation.main()  
fastr.examples.cross_validation.sink_data(network)  
fastr.examples.cross_validation.source_data(network)
```

**elastix\_nipype Module**

```
fastr.examples.elastix_nipype.create_network()  
fastr.examples.elastix_nipype.main()  
fastr.examples.elastix_nipype.sink_data(network)  
fastr.examples.elastix_nipype.source_data(network)
```

**expand Module**

```
fastr.examples.expand.create_network()  
fastr.examples.expand.main()  
fastr.examples.expand.sink_data(network)  
fastr.examples.expand.source_data(network)
```

### **failing\_network Module**

```
fastr.examples.failing_network.create_network()  
fastr.examples.failing_network.main()  
fastr.examples.failing_network.sink_data(network)  
fastr.examples.failing_network.source_data(network)
```

### **filecopy Module**

```
fastr.examples.filecopy.create_network()  
fastr.examples.filecopy.main()  
fastr.examples.filecopy.sink_data(network)  
fastr.examples.filecopy.source_data(network)
```

### **macro\_node Module**

```
fastr.examples.macro_node.create_macro_network()  
fastr.examples.macro_node.create_network()  
fastr.examples.macro_node.create_super_macro_node()  
fastr.examples.macro_node.main()  
fastr.examples.macro_node.sink_data(network)  
fastr.examples.macro_node.source_data(network)
```

### **shift\_links Module**

```
fastr.examples.shift_links.create_network()  
fastr.examples.shift_links.main()  
fastr.examples.shift_links.sink_data(network)  
fastr.examples.shift_links.source_data(network)
```

### **source\_sink Module**

```
fastr.examples.source_sink.create_network()  
fastr.examples.source_sink.main()  
fastr.examples.source_sink.sink_data(network)  
fastr.examples.source_sink.source_data(network)
```

## **Subpackages**

### **test Package**

#### **test\_examples Module**

## execution Package

### execution Package

This package contains all modules related directly to the execution

#### 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\_\_** = dict\_proxy({'load': <function load>, '\_\_module\_\_': 'fastr.execution.environmentmodules', '\_\_repr\_\_': <...>})

**\_\_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\_\_** ()

**\_\_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

**avail\_modules**

List of avaible modules

**clear** ()

Unload all modules (except the protected modules as they cannot be unloaded). This should result in a clean environment.

**exception** = FastrImportError('Cannot find Environment Modules home directory (environment variables not set)')

**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

**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
Bases: enum.Enum

__format__(format_spec)
__module__ = 'fastr.execution.environmentmodules'

static __new__(value)
__reduce_ex__(proto)
__repr__()
__str__()
envmod = 'environmentmodules'
lmod = 'Lmod'
```

## executionpluginmanager Module

This module holds the ExecutionPluginManager as well as the base-class for all ExecutionPlugins.

```
class fastr.execution.executionpluginmanager.ExecutionPlugin(finished_callback=None,
                                                               can-
                                                               celled_callback=None,
                                                               sta-
                                                               tus_callback=None)
```

Bases: *fastr.core.baseplugin.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 with 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 traverse the dependency graph and kill each job manual.

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

#### **`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 with 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(['cleanup', '_queue_job', '__init__'])`

```
__del__()
    Cleanup if the variable was deleted on purpose

__enter__()
__exit__(type_, value, tb)

__init__(finished_callback=None, cancelled_callback=None, status_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.execution.executionpluginmanager'

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)

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

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

class fastr.execution.executionpluginmanager.ExecutionPluginManager
    Bases: fastr.core.pluginmanager.PluginSubManager
    Container holding all the ExecutionPlugins known to the Fastr system
    __abstractmethods__ = frozenset([()])
    __init__()
        Initialize a ExecutionPluginManager and load plugins.
        Parameters
            • path – path to search for plugins
            • recursive – flag for searching recursively
        Returns newly created ExecutionPluginManager
    __module__ = ‘fastr.execution.executionpluginmanager’

class fastr.execution.executionpluginmanager.JobAction
    Bases: enum.Enum
    Job actions that can be performed. This is used for checking if held jobs should be queued, held longer or be cancelled.
    __format__(format_spec)
    __module__ = ‘fastr.execution.executionpluginmanager’
    static __new__(value)
    __reduce_ex__(proto)
    __repr__()
    __str__()
    cancel = ‘cancel’
    hold = ‘hold’
    queue = ‘queue’
```

## 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.

## job Module

This module contains the Job class and some related classes.

```
class fastr.execution.job.InlineJob(*args, **kwargs)
    Bases: fastr.execution.job.Job

    __init__(*args, **kwargs)
        __module__ = 'fastr.execution.job'

    get_result()

class fastr.execution.job.Job(node, sample_id, sample_index, input_arguments, output_arguments, hold_jobs=None, status_callback=None, preferred_types=None)
    Bases: fastr.core.serializable.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__.pickle.gz'
```

```
INFO_DUMP = '__fastr_extra_job_info__.json'
```

```
PROV_DUMP = '__fastr_prov__.json'
```

```
RESULT_DUMP = '__fastr_result__.pickle.gz'
```

```
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, status_callback=None, preferred_types=None)
```

Create a job

**Parameters**

- **node** ([Node](#)) – the node the job is based on
- **sample\_id** ([fastr.core.samples.SampleId](#)) – the id of the sample
- **sample\_index** ([fastr.core.samples.SampleIndex](#)) – the index of the sample
- **input\_arguments** ([list\[dict\]](#)) – the argument list
- **output\_arguments** ([list\[dict\]](#)) – the argument list
- **hold\_jobs** ([list\[str\]](#)) – the jobs on which this jobs depend
- **status\_callback** ([callable](#)) – The callback to call when the status changed
- **preferred\_types** – 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 calc\_cardinality(description, payload)****commandfile**

The path of the command pickle

**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*

**extrainfofile**

The path where the extra job info document is saved

**extrainfourl**

The url where the extra job info document is saved

**fill\_output\_argument(output\_spec, cardinality, desired\_type, requested)**

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*

**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** *BaseDataType*

**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

**id**

The id of this job

**logfile**

The path of the result pickle

**logurl**

The url of the result pickle

**provfile**

The path where the prov document is saved

**provurl**

The url where the prov document is saved

**required\_cores**

Number of required cores

**required\_memory**

Number of required memory

**required\_time**

Number of required runtime

**status**

The status of the job

**stderrfile**

The path where the stderr text is saved

**stderrurl**

The url where the stderr text is saved

**stdoutfile**

The path where the stdout text is saved

**stdouturl**

The url where the stdout text is saved

**tmpurl**

The URL of the tmpdir to use

```
tool

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

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

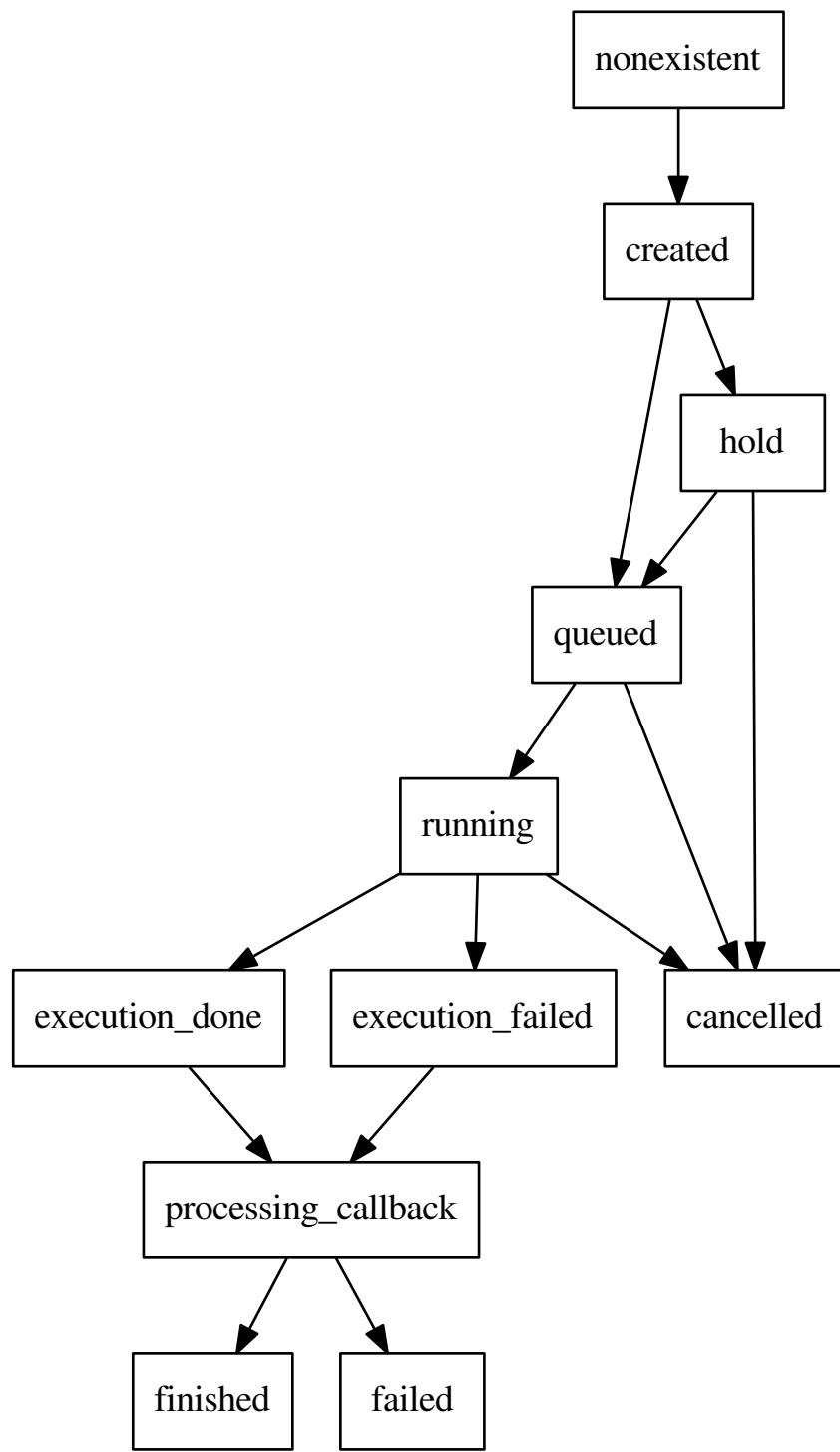
validate_results (payload)
    Validate the results of the Job

    Returns flag indicating the results are complete and valid

write()

class fastr.execution.job.JobState (_ , stage, error)
Bases: enum.Enum

The possible states a Job can be in. An overview of the states and the advised transitions are depicted in the following figure:
```



```
__format__(format_spec)
__init__(_, stage, error)
__module__ = 'fastr.execution.job'
static __new__(value)
```

```

__reduce_ex__(proto)
__repr__()
__str__()
cancelled = ('cancelled', 'done', True)
created = ('created', 'idle', False)
done
execution_done = ('execution_done', 'in_progress', False)
execution_failed = ('execution_failed', 'in_progress', True)
failed = ('failed', 'done', True)
finished = ('finished', 'done', False)
hold = ('hold', 'idle', False)
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, status_callback=None, preferred_types=None)
Bases: fastr.execution.job.Job

Special SinkJob for the Sink

__init__(node, sample_id, sample_index, input_arguments, output_arguments, hold_jobs=None, substitutions=None, status_callback=None, preferred_types=None)
__module__ = 'fastr.execution.job'

__repr__()
String representation for the SinkJob

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

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

**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(node, sample_id, sample_index, input_arguments,
                                      output_arguments, hold_jobs=None, status_callback=None, preferred_types=None)
```

Bases: *fastr.execution.job.Job*

Special SourceJob for the Source

**\_\_module\_\_** = ‘fastr.execution.job’

**\_\_repr\_\_**()

String representation for the SourceJob

**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

## networkanalyzer Module

Module that defines the NetworkAnalyzer and holds the reference implementation.

```
class fastr.execution.networkanalyzer.DefaultNetworkAnalyzer
```

Bases: *fastr.execution.networkanalyzer.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\_\_** = dict\_proxy({‘\_\_dict\_\_’: <attribute ‘\_\_dict\_\_’ of ‘NetworkAnalyzer’ objects>, ‘\_\_weakref\_\_’: <attribute

**\_\_module\_\_** = ‘fastr.execution.networkanalyzer’}

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**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: *fastr.execution.networkchunker.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'****chunck\_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\_\_ = dict\_proxy({‘\_\_dict\_\_’: <attribute ‘\_\_dict\_\_’ of ‘NetworkChunker’ objects>, ‘\_\_module\_\_’: ‘fastr.execu****\_\_module\_\_ = ‘fastr.execution.networkchunker’****\_\_weakref\_\_**

list of weak references to the object (if defined)

**chunck\_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

**resources Package****resources Package****Subpackages**

**datatypes Package**

**datatype Package**

**AnalyzeImageFile Module**

**Boolean Module**

**Directory Module**

**FilePrefix Module**

**Float Module**

**ITKImageFile Module**

**Int Module**

**JsonFile Module**

**MetaImageFile Module**

**NiftiImageFile Module**

**NiftiImageFileCompressed Module**

**NiftiImageFileUncompressed Module**

**Number Module**

**ProvNFile Module**

**String Module**

**TifImageFile Module**

**TxtFile Module**

**UnsignedInt Module**

**tools Package**

**tools Package**

**test Package**

**test Package**

**testdataatypes Module**

## utils Package

### utils Package

A collections of utils for fastr (command line tools or non-core functionality)

#### checksum Module

This module contains a number of functions for checksumming files and objects

`fastr.utils.checksum.checksum(filepath, algorithm='md5', hasher=None)`

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.utils.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.utils.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.utils.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.utils.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.utils.classproperty.ClassPropertyDescriptor (fget)`  
Bases: `object`

A descriptor that can act like a property for a class.

`__dict__ = dict_proxy({‘__module__’: ‘fastr.utils.classproperty’, ‘__dict__’: <attribute ‘__dict__’ of ‘ClassPropertyDescriptor’ objects>, ‘__weakref__’: <list of weak references to the object (if defined)>})`

`__get__(obj, cls=None)`

`__init__(fget)`

`__module__ = ‘fastr.utils.classproperty’`

`__weakref__`

`fastr.utils.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.utils.clear_pycs.dir_list(directory)`  
Find all .pyc files

**Parameters** `directory (str)` – directory to search

**Returns** all .pyc files

**Return type** list

`fastr.utils.clear_pycs.main()`  
Main entry point

### `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)`

### `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 other dictionary

**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()
```

**iohelpers Module**

```
fastr.utils.iohelpers.load_gpickle(path, retry_scheme=None)
```

```
fastr.utils.iohelpers.load_json(path)
```

```
fastr.utils.iohelpers.save_gpickle(path, data)
```

```
fastr.utils.iohelpers.save_json(path, data, indent=2)
```

**jsonschema parser 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.utils.jsonschemaparser.FastrRefResolver(base_uri, referrer, store=(),  
                                                 cache_remote=True, handlers=())
```

Bases: jsonschema.validators.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.utils.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.utils.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.utils.jsonschemaparser.extend validator_cls)
```

Extend the given jsonschema.IValidator with the Seep layer.

```
fastr.utils.jsonschemaparser.getblueprinter(uri, blueprint=None)
```

Instantiate the given data using the blueprinter.

**Parameters** **blueprint** – a blueprint (JSON Schema with Seep properties)

```
fastr.utils.jsonschemaparser.items_validate 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.utils.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.utils.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.utils.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.utils.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.utils.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

### **multiprocesswrapper Module**

```
fastr.utils.multiprocesswrapper.function_wrapper(filepath,    fnc_name,    *args,  
                                         **kwargs)
```

### **pim\_publisher Module**

```
class fastr.utils.pim_publisher.PimPublisher(uri=None)
```

Bases: `object`

Class to publish to PIM

```
PIM_STATUS_MAPPING = {<JobState.execution_done: ('execution_done', 'in_progress', False)>: 'running', <JobSta
```

```
__dict__ = dict_proxy({'__dict__': <attribute '__dict__' of 'PimPublisher' objects>, '__module__': 'fastr.utils.pim_
```

```
__init__(uri=None)
```

```
__module__ = 'fastr.utils.pim_publisher'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
pim_register_run(network)
```

```
static pim_serialize_network(network)
```

Serialize Network in the correct for to use with PIM.

**Returns** json data for PIM

```
pim_update_status(job)
```

### **procutils Module**

```
fastr.utils.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

### **rest\_generation Module**

```
fastr.utils.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`

## schematatable Module

A module to generate reStructuredText tables from json schema files

**class** `fastr.utils.schematatable.SchemaPrinter`(*schema*, *skipfirst=False*)

Bases: `object`

Object that create a table in reStructuedText from a json schema

**\_\_dict\_\_** = `dict_proxy({‘parse’: <function parse>, ‘__module__’: ‘fastr.utils.schematatable’, ‘descend’: <function descend>})`

**\_\_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.utils.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`

## sysinfo Module

This module contains function to help gather system information use for the provenance of the Job execution.

`fastr.utils.sysinfo.get_cpu_usage()`

Get the current CPU usage

**Returns** CPU usage info

**Return type** `dict`

`fastr.utils.sysinfo.get_drmaa_info()`

Get information about the SGE cluster (if applicable)

**Returns** cluster info

**Return type** `dict`

`fastr.utils.sysinfo.get_hostinfo()`

Get all information about the current host machine

**Returns** host info

**Return type** `dict`

`fastr.utils.sysinfo.get_memory_usage()`

Get the current memory usage

**Returns** memory usage info

**Return type** dict

`fastr.utils.sysinfo.get_mounts()`

Get the current mounts known on the system

**Returns** mount info

**Return type** dict

`fastr.utils.sysinfo.get_os()`

Get information about the OS

**Returns** OS information

**Return type** dict

`fastr.utils.sysinfo.get_processes()`

Get a list of all currently running processes

**Returns** process information

**Return type** list

`fastr.utils.sysinfo.get_python()`

Get information about the currently used Python implementation

**Returns** python info

**Return type** dict

`fastr.utils.sysinfo.get_sysinfo()`

Get system information (cpu, memory, mounts and users)

**Returns** system information

**Return type** dict

`fastr.utils.sysinfo.get_users()`

Get current users on the system

**Returns** user info

**Return type** dict

`fastr.utils.sysinfo.namedtuple_to_dict(ntuple)`

Helper function to convert a namedtuple into a dict

**Parameters** ntuple (namedtuple) – the namedtuple to convert

**Returns** named tuple as a dict

**Return type** dict

## verify Module

`fastr.utils.verify.verify_resource_loading(filename)`

`fastr.utils.verify.verify_tool(filename)`

Verify that a file

## xmltodict Module

This module contains tool for converting python dictionaries into XML object and vice-versa.

`fastr.utils.xmltodict.dump (data, filehandle)`

Write a dict to an XML file

### Parameters

- **data** – data to write
- **filehandle** – file handle to write to

`fastr.utils.xmltodict.dumps (data)`

Write a dict to an XML string

### Parameters **data** – data to write

### Returns the XML data

### Return type `str`

`fastr.utils.xmltodict.load (filehandle)`

Load an xml file and parse it to a dict

### Parameters **filehandle** – file handle to load

### Returns the parsed data

`fastr.utils.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

## Subpackages

### cmd Package

#### cmd Package

`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

### 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\_build')

```
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

### prov Module

```
fastr.utils.cmd.prov.get_parser()
```

```
fastr.utils.cmd.prov.get_prov_document(result)
```

```
fastr.utils.cmd.prov.main()
```

Get PROV information from the result pickle.

### 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

### testtool Module

```
fastr.utils.cmd.testtool.get_parser()
```

```
fastr.utils.cmd.testtool.main()
```

Run the tests of a tool to verify the proper function

### trace Module

```
fastr.utils.cmd.trace.get_parser()
```

```
fastr.utils.cmd.trace.main()
```

Trace samples/sinks from a run

```
fastr.utils.cmd.trace.print_job_result(job_file)
```

```
fastr.utils.cmd.trace.print_sample_node(sink_data, dirname, sample_node_tuples, verbose)
```

---

```
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)
```

**verify Module**

```
fastr.utils.cmd.verify.get_parser()
fastr.utils.cmd.verify.main()
    Print information from a job file
```

**webapp Module**

```
fastr.utils.cmd.webapp.get_parser()
fastr.utils.cmd.webapp.main()
    Start the fastr webapp and open in a new browser tab
fastr.utils.cmd.webapp.open_url(url)
```

**web Package****web Package****api Module**

```
class fastr.web.api.NetworkApi(api=None, *args, **kwargs)
    Bases: flask_restplus.resource.Resource
    __module__ = 'fastr.web.api'
    endpoint = 'api_network'
    get(id_)
        Get a Network json description from the server
    mediatypes(resource_cls)
    methods = ['GET']

class fastr.web.api.NetworkListApi(api=None, *args, **kwargs)
    Bases: flask_restplus.resource.Resource
    __module__ = 'fastr.web.api'
    endpoint = 'api_networks'
    get(*args, **kwargs)
        Get a list of the networks
    mediatypes(resource_cls)
    methods = ['GET']

class fastr.web.api.ObjectUrl(object_classs, **kwargs)
    Bases: flask_restplus.fields.Raw
    __init__(object_classs, **kwargs)
```

```
__module__ = 'fastr.web.api'
__schema_type__ = 'string'
__slotnames__ = []
format (value)

class fastr.web.api.Run (id_, network, source_data, sink_data)
Bases: object

__dict__ = dict_proxy({'status': <function status>, '__module__': 'fastr.web.api', 'abort': <function abort>, '__di...'})
__init__ (id_, network, source_data, sink_data)
__module__ = 'fastr.web.api'
__weakref__
    list of weak references to the object (if defined)

abort ()
run_network (network, source_data, sink_data, abort_lock)

status ()

class fastr.web.api.RunApi (api=None, *args, **kwargs)
Bases: flask_restplus.resource.Resource

Run API documentation

__module__ = 'fastr.web.api'
delete (id_)
    Abort a Network run and stop all associated execution

endpoint = 'api_run'
get (*args, **kwargs)
    Get information about a Network run

mediatypes (resource_cls)
methods = ['DELETE', 'GET']

class fastr.web.api.RunListApi (api=None, *args, **kwargs)
Bases: flask_restplus.resource.Resource

__module__ = 'fastr.web.api'
endpoint = 'api_runs'
get (*args, **kwargs)
    Get a list of all Network runs on the server

mediatypes (resource_cls)
methods = ['GET', 'POST']

post ()
    Create a new Network run and start execution

request_parser = <flask_restplus.reqparse.RequestParser object>

class fastr.web.api.StatusApi (api=None, *args, **kwargs)
Bases: flask_restplus.resource.Resource

__module__ = 'fastr.web.api'
endpoint = 'api_status'
get (id_)
    Get the status of a Network Run on the server
```

```

mediatypes (resource_cls)
methods = ['GET']

class fastr.web.api.SubUrl (object_classs, subfield, **kwargs)
    Bases: flask_restplus.fields.Raw

        __init__ (object_classs, subfield, **kwargs)
        __module__ = 'fastr.web.api'
        __schema_type__ = 'string'
        format (value)

class fastr.web.api.ToolApi (api=None, *args, **kwargs)
    Bases: flask_restplus.resource.Resource

        __module__ = 'fastr.web.api'
        endpoint = 'api_version_tool'
        get (id_, version=None)
            Get a Tool json description from the server
        mediatypes (resource_cls)
        methods = ['GET']

class fastr.web.api.ToolListApi (api=None, *args, **kwargs)
    Bases: flask_restplus.resource.Resource

        __module__ = 'fastr.web.api'
        endpoint = 'api_tools'
        get (*args, **kwargs)
            Get a list of all Tools known to the server
        mediatypes (resource_cls)
        methods = ['GET']

        fastr.web.api.network_lock_thread (lock, network)
        fastr.web.api.network_runner (network, source_data, sink_data, chuck_status, job_status,
                                    job_results, abort_lock)
        fastr.web.api.update_job_result (job, job_status, job_results)
        fastr.web.api.update_status (job, job_status)

```

## run Module

```

fastr.web.run.main ()
fastr.web.run.runapp (debug=False)

```

## views Module

```

fastr.web.views.doc ()
fastr.web.views.index ()
fastr.web.views.network (name=None)
fastr.web.views.networks ()
fastr.web.views.prov ()

```

```
fastr.web.views.shutdown()  
fastr.web.views.shutdown_server()  
fastr.web.views.tool(toolname=None, version=None)  
fastr.web.views.websocket_client()
```

# CHAPTER 5

---

## Indices and tables

---

- genindex
- modindex
- search



---

## HTTP Routing Table

---

### /api

```
GET /api/doc, 52
GET /api/networks, 51
GET /api/networks/(id), 52
GET /api/runs, 52
GET /api/runs/(id), 52
GET /api/runs/(id)/status, 52
GET /api/tools, 52
GET /api/tools/(id), 52
GET /api/tools/(id)/(version), 52
POST /api/runs, 52
DELETE /api/runs/(id), 52
```

### /static

```
GET /static/(path:filename), 52
```

### /swagger.json

```
GET /swagger.json, 52
```

### /swaggerui

```
GET /swaggerui/(path:filename), 52
```



---

## Python Module Index

---

### f

fastr.`__init__`, 53  
fastr.configmanager, 64  
fastr.core, 96  
fastr.core.basemanager, 96  
fastr.core.baseplugin, 98  
fastr.core.datatypemanager, 99  
fastr.core.dimension, 101  
fastr.core.inputoutput, 102  
fastr.core.interface, 115  
fastr.core.ioplugin, 117  
fastr.core.link, 120  
fastr.core.network, 123  
fastr.core.networkmanager, 126  
fastr.core.node, 127  
fastr.core.objectmanager, 138  
fastr.core.pluginmanager, 139  
fastr.core.provenance, 141  
fastr.core.samples, 142  
fastr.core.serializable, 147  
fastr.core.target, 150  
fastr.core.test, 158  
fastr.core.tool, 152  
fastr.core.toolmanager, 154  
fastr.core.updateable, 155  
fastr.core.version, 157  
fastr.core.vfs, 158  
fastr.data, 159  
fastr.data.url, 159  
fastr.datatypes, 66  
fastr.examples, 160  
fastr.examples.`__generate_reference__`, 160  
fastr.examples.add\_ints, 160  
fastr.examples.auto\_prefix, 161  
fastr.examples.collapse, 161  
fastr.examples.collapse\_expand, 161  
fastr.examples.cross\_validation, 161  
fastr.examples.elastix\_nipype, 161  
fastr.examples.expand, 161  
fastr.examples.failing\_network, 162  
fastr.examples.filecopy, 162  
fastr.examples.macro\_node, 162  
fastr.examples.shift\_links, 162  
fastr.examples.source\_sink, 162  
fastr.exceptions, 75  
fastr.execution, 163  
fastr.execution.environmentmodules, 163  
fastr.execution.executionpluginmanager, 164  
fastr.execution.executionscript, 167  
fastr.execution.job, 168  
fastr.execution.networkanalyzer, 174  
fastr.execution.networkchunker, 175  
fastr.plugins, 82  
fastr.resources, 175  
fastr.utils, 177  
fastr.utils.checksum, 177  
fastr.utils.classproperty, 178  
fastr.utils.clear\_pycs, 178  
fastr.utils.cmd, 185  
fastr.utils.cmd.cat, 185  
fastr.utils.cmd.execute, 186  
fastr.utils.cmd.extract\_argparse, 186  
fastr.utils.cmd.prov, 186  
fastr.utils.cmd.run, 186  
fastr.utils.cmd.testtool, 186  
fastr.utils.cmd.trace, 186  
fastr.utils.cmd.verify, 187  
fastr.utils.cmd.webapp, 187  
fastr.utils.compare, 178  
fastr.utils.dicteq, 178  
fastr.utils.gettools, 179  
fastr.utils.iohelpers, 179  
fastr.utils.jsonschemasparser, 179  
fastr.utils.multiprocesswrapper, 182  
fastr.utils.pim\_publisher, 182  
fastr.utils.procutils, 182  
fastr.utils.rest\_generation, 182  
fastr.utils.schematotable, 183  
fastr.utils.sysinfo, 183  
fastr.utils.verify, 184  
fastr.utils.xmltodict, 185  
fastr.version, 95  
fastr.web, 187  
fastr.web.api, 187

`fastr.web.run`, 189  
`fastr.web.views`, 189

### Symbols

- \_\_abstractmethods\_\_ (fastr.\_\_init\_\_.ConstantNode attribute), 62
- \_\_abstractmethods\_\_ (fastr.\_\_init\_\_.Link attribute), 57
- \_\_abstractmethods\_\_ (fastr.\_\_init\_\_.Node attribute), 60
- \_\_abstractmethods\_\_ (fastr.\_\_init\_\_.SourceNode attribute), 63
- \_\_abstractmethods\_\_ (fastr.core.basemanager.BaseManager attribute), 96
- \_\_abstractmethods\_\_ (fastr.core.baseplugin.BasePlugin attribute), 98
- \_\_abstractmethods\_\_ (fastr.core.baseplugin.Plugin attribute), 99
- \_\_abstractmethods\_\_ (fastr.core.datatypemanager.DataTypeManager attribute), 99
- \_\_abstractmethods\_\_ (fastr.core.dimension.HasDimensions attribute), 102
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.AdvancedFlowOutput attribute), 102
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.BaseInput attribute), 102
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.BaseInputOutput attribute), 103
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.BaseOutput attribute), 105
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.Input attribute), 105
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.Output attribute), 108
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.SourceOutput attribute), 110
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.SubInput attribute), 111
- \_\_abstractmethods\_\_ (fastr.core.inputoutput.SubOutput attribute), 113
- \_\_abstractmethods\_\_ (fastr.core.interface.Interface attribute), 115
- \_\_abstractmethods\_\_ (fastr.core.interface.InterfacePluginManager attribute), 116
- \_\_abstractmethods\_\_ (fastr.core.ioplugin.IOPlugin attribute), 117
- \_\_abstractmethods\_\_ (fastr.core.ioplugin.IOPPluginManager attribute), 119
- \_\_abstractmethods\_\_ (fastr.core.link.Link attribute), 120
- \_\_abstractmethods\_\_ (fastr.core.networkmanager.NetworkManager attribute), 126
- \_\_abstractmethods\_\_ (fastr.core.node.AdvancedFlowNode attribute), 127
- \_\_abstractmethods\_\_ (fastr.core.node.ConstantNode attribute), 127
- \_\_abstractmethods\_\_ (fastr.core.node.FlowNode attribute), 129
- \_\_abstractmethods\_\_ (fastr.core.node.InputGroup attribute), 130
- \_\_abstractmethods\_\_ (fastr.core.node.MacroNode attribute), 132
- \_\_abstractmethods\_\_ (fastr.core.node.Node attribute), 132
- \_\_abstractmethods\_\_ (fastr.core.node.SinkNode attribute), 135
- \_\_abstractmethods\_\_ (fastr.core.node.SourceNode attribute), 137
- \_\_abstractmethods\_\_ (fastr.core.objectmanager.ObjectManager attribute), 138
- \_\_abstractmethods\_\_ (fastr.core.pluginmanager.BasePluginManager attribute), 139
- \_\_abstractmethods\_\_ (fastr.core.pluginmanager.PluginManager attribute), 140
- \_\_abstractmethods\_\_ (fastr.core.pluginmanager.PluginSubManager attribute), 140
- \_\_abstractmethods\_\_ (fastr.core.pluginmanager.PluginsView attribute), 140
- \_\_abstractmethods\_\_ (fastr.core.samples.HasSamples attribute), 142
- \_\_abstractmethods\_\_ (fastr.core.samples.SampleCollection attribute), 143
- \_\_abstractmethods\_\_ (fastr.core.samples.SampleValue attribute), 147
- \_\_abstractmethods\_\_ (fastr.core.target.DockerTarget attribute), 150
- \_\_abstractmethods\_\_ (fastr.core.target.LocalBinaryTarget attribute), 150
- \_\_abstractmethods\_\_ (fastr.core.target.ProcessUsageCollection attribute), 151
- \_\_abstractmethods\_\_ (fastr.core.target.Target attribute), 152

\_\_abstractmethods\_\_ (fastr.core.toolmanager.ToolManager attribute), 154  
\_\_abstractmethods\_\_ (fastr.core.updateable.Updateable attribute), 155  
\_\_abstractmethods\_\_ (fastr.datatypes.AnalyzeImageFile attribute), 66  
\_\_abstractmethods\_\_ (fastr.datatypes.AnyFile attribute), 66  
\_\_abstractmethods\_\_ (fastr.datatypes.AnyType attribute), 66  
\_\_abstractmethods\_\_ (fastr.datatypes.BaseDataType attribute), 67  
\_\_abstractmethods\_\_ (fastr.datatypes.Boolean attribute), 68  
\_\_abstractmethods\_\_ (fastr.datatypes.DataType attribute), 68  
\_\_abstractmethods\_\_ (fastr.datatypes.Deferred attribute), 69  
\_\_abstractmethods\_\_ (fastr.datatypes.Directory attribute), 70  
\_\_abstractmethods\_\_ (fastr.datatypes.EnumType attribute), 70  
\_\_abstractmethods\_\_ (fastr.datatypes.FilePrefix attribute), 70  
\_\_abstractmethods\_\_ (fastr.datatypes.Float attribute), 71  
\_\_abstractmethods\_\_ (fastr.datatypes.ITKImageFile attribute), 71  
\_\_abstractmethods\_\_ (fastr.datatypes.Int attribute), 71  
\_\_abstractmethods\_\_ (fastr.datatypes.JsonFile attribute), 71  
\_\_abstractmethods\_\_ (fastr.datatypes.MetaImageFile attribute), 72  
\_\_abstractmethods\_\_ (fastr.datatypes.NiftiImageFile attribute), 72  
\_\_abstractmethods\_\_ (fastr.datatypes.NiftiImageFileCompressed attribute), 72  
\_\_abstractmethods\_\_ (fastr.datatypes.NiftiImageFileUncompressed attribute), 72  
\_\_abstractmethods\_\_ (fastr.datatypes.Number attribute), 72  
\_\_abstractmethods\_\_ (fastr.datatypes.ProvNFile attribute), 73  
\_\_abstractmethods\_\_ (fastr.datatypes.String attribute), 73  
\_\_abstractmethods\_\_ (fastr.datatypes.TifImageFile attribute), 73  
\_\_abstractmethods\_\_ (fastr.datatypes.TxtFile attribute), 73  
\_\_abstractmethods\_\_ (fastr.datatypes.TypeGroup attribute), 73  
\_\_abstractmethods\_\_ (fastr.datatypes.URLType attribute), 74  
\_\_abstractmethods\_\_ (fastr.datatypes.UnsignedInt attribute), 75  
\_\_abstractmethods\_\_ (fastr.datatypes.ValueType attribute), 75  
\_\_abstractmethods\_\_ (fastr.execution.executionpluginmanager.Plugin attribute), 165  
\_\_abstractmethods\_\_ (fastr.execution.executionpluginmanager.Execution attribute), 167  
\_\_abstractmethods\_\_ (fastr.plugins.BlockingExecution attribute), 82  
\_\_abstractmethods\_\_ (fastr.plugins.CommaSeperatedValueFile attribute), 83  
\_\_abstractmethods\_\_ (fastr.plugins.CrossValidation attribute), 83  
\_\_abstractmethods\_\_ (fastr.plugins.DRMAAExecution attribute), 83  
\_\_abstractmethods\_\_ (fastr.plugins.FastrInterface attribute), 84  
\_\_abstractmethods\_\_ (fastr.plugins.FileSystem attribute), 85  
\_\_abstractmethods\_\_ (fastr.plugins.FlowInterface attribute), 86  
\_\_abstractmethods\_\_ (fastr.plugins.LinearExecution attribute), 87  
\_\_abstractmethods\_\_ (fastr.plugins.NipypeInterface attribute), 87  
\_\_abstractmethods\_\_ (fastr.plugins.Null attribute), 88  
\_\_abstractmethods\_\_ (fastr.plugins.ProcessPoolExecution attribute), 89  
\_\_abstractmethods\_\_ (fastr.plugins.RQExecution attribute), 89  
\_\_abstractmethods\_\_ (fastr.plugins.Reference attribute), 89  
\_\_abstractmethods\_\_ (fastr.plugins.VirtualFileSystem attribute), 90  
\_\_abstractmethods\_\_ (fastr.plugins.VirtualFileSystemRegularExpression attribute), 92  
\_\_abstractmethods\_\_ (fastr.plugins.VirtualFileSystemValueList attribute), 93  
\_\_abstractmethods\_\_ (fastr.plugins.XNATStorage attribute), 94  
\_\_add\_\_() (fastr.configmanager.EmptyDefault method), 66  
\_\_add\_\_() (fastr.core.samples.SampleBaseId method), 142  
\_\_add\_\_() (fastr.core.samples.SampleItemBase method), 145  
\_\_add\_\_() (fastr.core.samples.SamplePayload method), 146  
\_\_add\_\_() (fastr.core.samples.SampleValue method), 147  
\_\_contains\_\_() (fastr.core.objectmanager.ObjectManager method), 138  
\_\_contains\_\_() (fastr.core.samples.HasSamples method), 142  
\_\_contains\_\_() (fastr.core.samples.SampleCollection method), 143  
\_\_dataschemafile\_\_ (fastr.\_\_init\_\_.ConstantNode attribute), 62  
\_\_dataschemafile\_\_ (fastr.\_\_init\_\_.Link attribute), 57  
\_\_dataschemafile\_\_ (fastr.\_\_init\_\_.Network attribute), 54  
\_\_dataschemafile\_\_ (fastr.execution.executionpluginmanager.Plugin attribute), 60

`__dataschemafile__(fastr.__init__.SourceNode attribute), 63`  
`__dataschemafile__(fastr.core.link.Link attribute), 120`  
`__dataschemafile__(fastr.core.network.Network attribute), 123`  
`__dataschemafile__(fastr.core.node.ConstantNode attribute), 127`  
`__dataschemafile__(fastr.core.node.Node attribute), 132`  
`__dataschemafile__(fastr.core.node.SinkNode attribute), 135`  
`__dataschemafile__(fastr.core.node.SourceNode attribute), 137`  
`__dataschemafile__(fastr.core.tool.Tool attribute), 152`  
`__dataschemafile__(fastr.plugins.FastrInterface attribute), 84`  
`__dataschemafile__(fastr.plugins.FlowInterface attribute), 86`  
`__del__(fastr.execution.executionpluginmanager.ExecutionPlugin method), 165`  
`__delitem__(fastr.configmanager.EmptyDefault method), 66`  
`__delitem__(fastr.core.basemanager.BaseManager method), 96`  
`__delitem__(fastr.core.node.InputGroup method), 130`  
`__delitem__(fastr.core.pluginmanager.PluginsView method), 140`  
`__delitem__(fastr.core.samples.SampleCollection method), 143`  
`__delitem__(fastr.core.samples.SampleValue method), 147`  
`__dict__(fastr.configmanager.Config attribute), 64`  
`__dict__(fastr.configmanager.EmptyDefault attribute), 66`  
`__dict__(fastr.core.baseplugin.BasePlugin attribute), 98`  
`__dict__(fastr.core.dimension.Dimension attribute), 101`  
`__dict__(fastr.core.dimension.HasDimensions attribute), 102`  
`__dict__(fastr.core.interface.InputSpec attribute), 115`  
`__dict__(fastr.core.interface.InterfaceResult attribute), 116`  
`__dict__(fastr.core.interface.OutputSpec attribute), 116`  
`__dict__(fastr.core.node.DefaultInputGroupCombiner attribute), 128`  
`__dict__(fastr.core.pluginmanager.plugin_option_type attribute), 141`  
`__dict__(fastr.core.provenance.Provenance attribute), 141`  
`__dict__(fastr.core.samples.HasSamples attribute), 142`  
`__dict__(fastr.core.samples.SampleBaseId attribute), 142`  
`__dict__(fastr.core.samples.SampleItemBase attribute), 145`  
`__dict__(fastr.core.serializable.PassThroughSerializer attribute), 147`  
`__dict__(fastr.core.serializable.Serializable attribute), 148`  
`__dict__(fastr.core.target.SystemUsageInfo attribute), 151`  
`__dict__(fastr.core.target.Target attribute), 152`  
`__dict__(fastr.core.updateable.Updateable attribute), 155`  
`__dict__(fastr.core.version.Version attribute), 157`  
`__dict__(fastr.execution.environmentmodules.EnvironmentModules attribute), 163`  
`__dict__(fastr.execution.networkanalyzer.NetworkAnalyzer attribute), 174`  
`__dict__(fastr.execution.networkchunker.NetworkChunker attribute), 175`  
`__dict__(fastr.utils.classproperty.ClassPropertyDescriptor attribute), 178`  
`__dict__(fastr.utils.pim_publisher.PimPublisher attribute), 182`  
`__dict__(fastr.utils.schematatable.SchemaPrinter attribute), 183`  
`__dict__(fastr.web.api.Run attribute), 188`  
`__enter__(fastr.core.target.DockerTarget method), 150`  
`__enter__(fastr.core.target.LocalBinaryTarget method), 150`  
`__enter__(fastr.core.target.Target method), 152`  
`__enter__(fastr.execution.executionpluginmanager.ExecutionPlugin method), 166`  
`__eq__(fastr.__init__.Link method), 57`  
`__eq__(fastr.__init__.Network method), 54`  
`__eq__(fastr.__init__.Node method), 60`  
`__eq__(fastr.__init__.SourceNode method), 63`  
`__eq__(fastr.core.inputoutput.Input method), 105`  
`__eq__(fastr.core.inputoutput.Output method), 108`  
`__eq__(fastr.core.inputoutput.SubInput method), 111`  
`__eq__(fastr.core.inputoutput.SubOutput method), 113`  
`__eq__(fastr.core.link.Link method), 120`  
`__eq__(fastr.core.network.Network method), 123`  
`__eq__(fastr.core.node.Node method), 132`  
`__eq__(fastr.core.node.SourceNode method), 137`  
`__eq__(fastr.core.tool.Tool method), 152`  
`__eq__(fastr.datatypes.BaseDataType method), 67`  
`__eq__(fastr.datatypes.Directory method), 70`  
`__eq__(fastr.datatypes.MetaImageFile method), 72`  
`__eq__(fastr.datatypes.URLType method), 74`  
`__eq__(fastr.plugins.FastrInterface method), 84`  
`__eq__(fastr.plugins.FlowInterface method), 86`  
`__eq__(fastr.plugins.NipypeInterface method), 87`  
`__exit__(fastr.core.target.DockerTarget method), 150`  
`__exit__(fastr.core.target.LocalBinaryTarget method), 150`  
`__exit__(fastr.core.target.Target method), 152`  
`__exit__(fastr.execution.executionpluginmanager.ExecutionPlugin method), 166`  
`__format__(fastr.core.baseplugin.PluginState`

method), 99  
\_\_format\_\_() (fastr.execution.environmentmodules.ModuleSystem method), 103  
    method), 164  
\_\_format\_\_() (fastr.execution.executionpluginmanager.JobAction method), 105  
    method), 167  
\_\_format\_\_() (fastr.execution.job.JobState method), 172  
\_\_get\_\_() (fastr.utils.classproperty.ClassPropertyDescriptor method), 178  
\_\_getattr\_\_() (fastr.core.pluginmanager.LazyModule method), 139  
\_\_getitem\_\_() (fastr.\_\_init\_\_.Link method), 58  
\_\_getitem\_\_() (fastr.\_\_init\_\_.Network method), 54  
\_\_getitem\_\_() (fastr.configmanager.EmptyDefault method), 66  
\_\_getitem\_\_() (fastr.core.basemanager.BaseManager method), 96  
\_\_getitem\_\_() (fastr.core.inputoutput.Input method), 105  
\_\_getitem\_\_() (fastr.core.inputoutput.Output method), 108  
\_\_getitem\_\_() (fastr.core.inputoutput.SourceOutput method), 110  
\_\_getitem\_\_() (fastr.core.inputoutput.SubInput method), 112  
\_\_getitem\_\_() (fastr.core.inputoutput.SubOutput method), 113  
\_\_getitem\_\_() (fastr.core.link.Link method), 121  
\_\_getitem\_\_() (fastr.core.network.Network method), 123  
\_\_getitem\_\_() (fastr.core.node.InputGroup method), 130  
\_\_getitem\_\_() (fastr.core.objectmanager.ObjectManager method), 138  
\_\_getitem\_\_() (fastr.core.pluginmanager.BasePluginManager method), 139  
\_\_getitem\_\_() (fastr.core.pluginmanager.PluginsView method), 140  
\_\_getitem\_\_() (fastr.core.samples.HasSamples method), 142  
\_\_getitem\_\_() (fastr.core.samples.SampleCollection method), 143  
\_\_getitem\_\_() (fastr.core.samples.SampleValue method), 147  
\_\_getitem\_\_() (fastr.core.target.ProcessUsageCollection method), 151  
\_\_getnewargs\_\_() (fastr.core.pluginmanager.plugin\_option\_type method), 141  
\_\_getnewargs\_\_() (fastr.core.samples.SampleItemBase method), 145  
\_\_getnewargs\_\_() (fastr.core.target.SystemUsageInfo method), 151  
\_\_getstate\_\_() (fastr.\_\_init\_\_.ConstantNode method), 62  
\_\_getstate\_\_() (fastr.\_\_init\_\_.Link method), 58  
\_\_getstate\_\_() (fastr.\_\_init\_\_.Network method), 54  
\_\_getstate\_\_() (fastr.\_\_init\_\_.Node method), 60  
\_\_getstate\_\_() (fastr.\_\_init\_\_.SourceNode method), 63  
\_\_getstate\_\_() (fastr.core.inputoutput.BaseInputOutput method), 103  
    \_\_getstate\_\_() (fastr.core.inputoutput.Input method), 105  
    \_\_getstate\_\_() (fastr.core.inputoutput.Output method), 108  
    \_\_getstate\_\_() (fastr.core.inputoutput.SubInput method), 112  
    \_\_getstate\_\_() (fastr.core.inputoutput.SubOutput method), 114  
\_\_getstate\_\_() (fastr.core.interface.Interface method), 115  
\_\_getstate\_\_() (fastr.core.link.Link method), 121  
\_\_getstate\_\_() (fastr.core.network.Network method), 123  
\_\_getstate\_\_() (fastr.core.node.ConstantNode method), 127  
\_\_getstate\_\_() (fastr.core.node.MacroNode method), 132  
\_\_getstate\_\_() (fastr.core.node.Node method), 133  
\_\_getstate\_\_() (fastr.core.node.SinkNode method), 135  
\_\_getstate\_\_() (fastr.core.node.SourceNode method), 137  
\_\_getstate\_\_() (fastr.core.pluginmanager.plugin\_option\_type method), 141  
\_\_getstate\_\_() (fastr.core.samples.SampleValue method), 147  
\_\_getstate\_\_() (fastr.core.serializable.Serializable method), 148  
\_\_getstate\_\_() (fastr.core.target.SystemUsageInfo method), 151  
\_\_getstate\_\_() (fastr.core.tool.Tool method), 152  
\_\_getstate\_\_() (fastr.core.updateable.Updateable method), 155  
\_\_getstate\_\_() (fastr.datatypes.BaseDataType method), 67  
\_\_getstate\_\_() (fastr.datatypes.Deferred method), 69  
\_\_getstate\_\_() (fastr.execution.job.Job method), 168  
\_\_getstate\_\_() (fastr.plugins.FastrInterface method), 84  
\_\_getstate\_\_() (fastr.plugins.FlowInterface method), 86  
\_\_getstate\_\_() (fastr.plugins.NipypeInterface method), 87  
\_\_iadd\_\_() (fastr.configmanager.EmptyDefault method), 66  
\_\_init\_\_() (fastr.\_\_init\_\_.ConstantNode method), 62  
\_\_init\_\_() (fastr.\_\_init\_\_.Link method), 58  
\_\_init\_\_() (fastr.\_\_init\_\_.Network method), 54  
\_\_init\_\_() (fastr.\_\_init\_\_.Node method), 60  
\_\_init\_\_() (fastr.\_\_init\_\_.SourceNode method), 63  
\_\_init\_\_() (fastr.configmanager.Config method), 64  
\_\_init\_\_() (fastr.configmanager.EmptyDefault method), 66  
\_\_init\_\_() (fastr.core.basemanager.BaseManager method), 97  
\_\_init\_\_() (fastr.core.baseplugin.BasePlugin method), 98  
\_\_init\_\_() (fastr.core.datatypemanager.DataTypeManager method), 99

`__init__(fastr.core.dimension.Dimension method), 101`  
`__init__(fastr.core.inputoutput.BaseInput method), 102`  
`__init__(fastr.core.inputoutput.BaseInputOutput method), 103`  
`__init__(fastr.core.inputoutput.BaseOutput method), 105`  
`__init__(fastr.core.inputoutput.Input method), 106`  
`__init__(fastr.core.inputoutput.Output method), 108`  
`__init__(fastr.core.inputoutput.SourceOutput method), 111`  
`__init__(fastr.core.inputoutput.SubInput method), 112`  
`__init__(fastr.core.inputoutput.SubOutput method), 114`  
`__init__(fastr.core.interface.InterfacePluginManager method), 116`  
`__init__(fastr.core.interface.InterfaceResult method), 116`  
`__init__(fastr.core.ioplugin.IOPlugin method), 117`  
`__init__(fastr.core.ioplugin.IOPluginManager method), 119`  
`__init__(fastr.core.link.Link method), 121`  
`__init__(fastr.core.network.Network method), 123`  
`__init__(fastr.core.node.ConstantNode method), 127`  
`__init__(fastr.core.node.DefaultInputGroupCombiner method), 128`  
`__init__(fastr.core.node.FlowNode method), 129`  
`__init__(fastr.core.node.InputGroup method), 130`  
`__init__(fastr.core.node.MacroNode method), 132`  
`__init__(fastr.core.node.MergingInputGroupCombiner method), 132`  
`__init__(fastr.core.node.Node method), 133`  
`__init__(fastr.core.node.SinkNode method), 135`  
`__init__(fastr.core.node.SourceNode method), 137`  
`__init__(fastr.core.objectmanager.ObjectManager method), 138`  
`__init__(fastr.core.pluginmanager.BasePluginManager method), 139`  
`__init__(fastr.core.pluginmanager.LazyModule method), 140`  
`__init__(fastr.core.pluginmanager.PluginManager method), 140`  
`__init__(fastr.core.pluginmanager.PluginSubManager method), 140`  
`__init__(fastr.core.pluginmanager.PluginsView method), 140`  
`__init__(fastr.core.provenance.Provenance method), 141`  
`__init__(fastr.core.samples.SampleCollection method), 143`  
`__init__(fastr.core.samples.SampleValue method), 147`  
`__init__(fastr.core.target.DockerTarget method), 150`  
`__init__(fastr.core.target.LocalBinaryTarget method), 150`  
`__init__(fastr.core.target.ProcessUsageCollection method), 151`  
`__init__(fastr.core.tool.Tool method), 152`  
`__init__(fastr.core.updateable.Updateable method), 155`  
`__init__(fastr.datatypes.BaseDataType method), 67`  
`__init__(fastr.datatypes.DataType method), 68`  
`__init__(fastr.datatypes.Deferred method), 69`  
`__init__(fastr.datatypes.EnumType method), 70`  
`__init__(fastr.datatypes.TypeGroup method), 73`  
`__init__(fastr.datatypes.URLType method), 74`  
`__init__(fastr.datatypes.ValueType method), 75`  
`__init__(fastr.exceptions.FastrError method), 76`  
`__init__(fastr.exceptions.FastrExecutableNotFoundError method), 77`  
`__init__(fastr.exceptions.FastrSerializationError method), 79`  
`__init__(fastr.execution.environmentmodules.EnvironmentModules method), 163`  
`__init__(fastr.execution.executionpluginmanager.ExecutionPlugin method), 166`  
`__init__(fastr.execution.executionpluginmanager.ExecutionPluginManager method), 167`  
`__init__(fastr.execution.job.InlineJob method), 168`  
`__init__(fastr.execution.job.Job method), 168`  
`__init__(fastr.execution.job.JobState method), 172`  
`__init__(fastr.execution.job.SinkJob method), 173`  
`__init__(fastr.execution.networkchunker.DefaultNetworkChunker method), 175`  
`__init__(fastr.plugins.BlockingExecution method), 82`  
`__init__(fastr.plugins.CommaSepreatedValueFile method), 83`  
`__init__(fastr.plugins.DRMAAExecution method), 83`  
`__init__(fastr.plugins.FastrInterface method), 84`  
`__init__(fastr.plugins.FileSystem method), 85`  
`__init__(fastr.plugins.FlowInterface method), 86`  
`__init__(fastr.plugins.LinearExecution method), 87`  
`__init__(fastr.plugins.NipypeInterface method), 87`  
`__init__(fastr.plugins.Null method), 88`  
`__init__(fastr.plugins.ProcessPoolExecution method), 89`  
`__init__(fastr.plugins.RQExecution method), 89`  
`__init__(fastr.plugins.Reference method), 89`  
`__init__(fastr.plugins.VirtualFileSystem method), 90`  
`__init__(fastr.plugins.VirtualFileSystemRegularExpression method), 92`  
`__init__(fastr.plugins.VirtualFileSystemValueList method), 93`  
`__init__(fastr.plugins.XNATStorage method), 94`  
`__init__(fastr.utils.classproperty.ClassPropertyDescriptor method), 178`  
`__init__(fastr.utils.jsonschemaparser.FastrRefResolver method), 180`  
`__init__(fastr.utils.pim_publisher.PimPublisher method), 182`  
`__init__(fastr.utils.schematatable.SchemaPrinter method), 183`

`__init__()` (fastr.web.api.ObjectUrl method), 187  
`__init__()` (fastr.web.api.Run method), 188  
`__init__()` (fastr.web.api.SubUrl method), 189  
`__iter__()` (fastr.core.basemanager.BaseManager method), 97  
`__iter__()` (fastr.core.inputoutput.BaseInputOutput method), 103  
`__iter__()` (fastr.core.ioplugin.IOPluginManager method), 119  
`__iter__()` (fastr.core.node.DefaultInputGroupCombiner method), 128  
`__iter__()` (fastr.core.pluginmanager.PluginsView method), 140  
`__iter__()` (fastr.core.samples.HasSamples method), 142  
`__iter__()` (fastr.core.samples.SampleCollection method), 143  
`__iter__()` (fastr.core.samples.SampleValue method), 147  
`__keytransform__()` (fastr.core.basemanager.BaseManager method), 97  
`__keytransform__()` (fastr.core.datatypemanager.DataTypeManager method), 99  
`__keytransform__()` (fastr.core.ioplugin.IOPluginManager method), 119  
`__keytransform__()` (fastr.core.objectmanager.ObjectManager method), 138  
`__len__()` (fastr.core.basemanager.BaseManager method), 97  
`__len__()` (fastr.core.inputoutput.SubOutput method), 114  
`__len__()` (fastr.core.pluginmanager.PluginsView method), 141  
`__len__()` (fastr.core.samples.SampleCollection method), 144  
`__len__()` (fastr.core.samples.SampleValue method), 147  
`__len__()` (fastr.core.target.ProcessUsageCollection method), 151  
`__lshift__()` (fastr.core.inputoutput.BaseInput method), 103  
`__metaclass__` (fastr.\_\_init\_\_.Node attribute), 60  
`__metaclass__` (fastr.core.basemanager.BaseManager attribute), 97  
`__metaclass__` (fastr.core.baseplugin.BasePlugin attribute), 98  
`__metaclass__` (fastr.core.dimension.HasDimensions attribute), 102  
`__metaclass__` (fastr.core.interface.Interface attribute), 116  
`__metaclass__` (fastr.core.ioplugin.IOPlugin attribute), 117  
`__metaclass__` (fastr.core.node.InputGroup attribute), 130  
`__metaclass__` (fastr.core.node.Node attribute), 133  
`__metaclass__` (fastr.core.samples.HasSamples attribute), 142  
`__metaclass__` (fastr.core.target.Target attribute), 152  
`__metaclass__` (fastr.core.updateable.Updateable attribute), 155  
`__module__` (fastr.\_\_init\_\_.ConstantNode attribute), 63  
`__module__` (fastr.\_\_init\_\_.Link attribute), 58  
`__module__` (fastr.\_\_init\_\_.Network attribute), 54  
`__module__` (fastr.\_\_init\_\_.Node attribute), 60  
`__module__` (fastr.\_\_init\_\_.SourceNode attribute), 64  
`__module__` (fastr.configmanager.Config attribute), 65  
`__module__` (fastr.configmanager.EmptyDefault attribute), 66  
`__module__` (fastr.core.basemanager.BaseManager attribute), 97  
`__module__` (fastr.core.baseplugin.BasePlugin attribute), 98  
`__module__` (fastr.core.baseplugin.Plugin attribute), 99  
`__module__` (fastr.core.baseplugin.PluginMeta attribute), 99  
`__module__` (fastr.core.baseplugin.PluginState attribute), 99  
`__module__` (fastr.core.datatypemanager.DataTypeManager attribute), 100  
`__module__` (fastr.core.dimension.Dimension attribute), 101  
`__module__` (fastr.core.dimension.HasDimensions attribute), 102  
`__module__` (fastr.core.inputoutput.AdvancedFlowOutput attribute), 102  
`__module__` (fastr.core.inputoutput.BaseInput attribute), 103  
`__module__` (fastr.core.inputoutput.BaseInputOutput attribute), 104  
`__module__` (fastr.core.inputoutput.BaseOutput attribute), 105  
`__module__` (fastr.core.inputoutput.Input attribute), 106  
`__module__` (fastr.core.inputoutput.Output attribute), 109  
`__module__` (fastr.core.inputoutput.SourceOutput attribute), 111  
`__module__` (fastr.core.inputoutput.SubInput attribute), 112  
`__module__` (fastr.core.inputoutput.SubOutput attribute), 114  
`__module__` (fastr.core.interface.InputSpec attribute), 115  
`__module__` (fastr.core.interface.Interface attribute), 116  
`__module__` (fastr.core.interface.InterfacePluginManager attribute), 116  
`__module__` (fastr.core.interface.InterfaceResult attribute), 116  
`__module__` (fastr.core.interface.OutputSpec attribute), 116  
`__module__` (fastr.core.ioplugin.IOPlugin attribute), 117  
`__module__` (fastr.core.ioplugin.IOPluginManager attribute), 119  
`__module__` (fastr.core.link.Link attribute), 121

\_\_module\_\_ (fastr.core.network.Network attribute), 123  
 \_\_module\_\_ (fastr.core.serializable.Serializable attribute), 148  
 \_\_module\_\_ (fastr.core.networkmanager.NetworkManager module\_\_ (fastr.core.target.DockerTarget attribute), attribute), 127 150  
 \_\_module\_\_ (fastr.core.node.AdvancedFlowNode attribute), 127  
 \_\_module\_\_ (fastr.core.node.ConstantNode attribute), 128  
 \_\_module\_\_ (fastr.core.node.DefaultInputGroupCombiner module\_\_ (fastr.core.target.ProcessUsageCollection attribute), 128 attribute), 151  
 \_\_module\_\_ (fastr.core.node.FlowNode attribute), 129  
 \_\_module\_\_ (fastr.core.node.InputDict attribute), 130  
 \_\_module\_\_ (fastr.core.node.InputGroup attribute), 130  
 \_\_module\_\_ (fastr.core.node.MacroNode attribute), 132  
 \_\_module\_\_ (fastr.core.node.MergingInputGroupCombiner attribute), 132  
 \_\_module\_\_ (fastr.core.node.Node attribute), 133  
 \_\_module\_\_ (fastr.core.node.OutputDict attribute), 135  
 \_\_module\_\_ (fastr.core.node.SinkNode attribute), 135  
 \_\_module\_\_ (fastr.core.node.SourceNode attribute), 137  
 \_\_module\_\_ (fastr.core.objectmanager.ObjectManager attribute), 138  
 \_\_module\_\_ (fastr.core.pluginmanager.BasePluginManager attribute), 139  
 \_\_module\_\_ (fastr.core.pluginmanager.LazyModule attribute), 140  
 \_\_module\_\_ (fastr.core.pluginmanager.PluginManager attribute), 140  
 \_\_module\_\_ (fastr.core.pluginmanager.PluginSubManager attribute), 140  
 \_\_module\_\_ (fastr.core.pluginmanager.PluginsView attribute), 141  
 \_\_module\_\_ (fastr.core.pluginmanager.plugin\_option\_type module\_\_ (fastr.datatypes.Int attribute), 71 attribute), 141  
 \_\_module\_\_ (fastr.core.provenance.Provenance attribute), 141  
 \_\_module\_\_ (fastr.core.samples.HasSamples attribute), 142  
 \_\_module\_\_ (fastr.core.samples.SampleBaseId attribute), 142  
 \_\_module\_\_ (fastr.core.samples.SampleCollection attribute), 144  
 \_\_module\_\_ (fastr.core.samples.SampleId attribute), 144  
 \_\_module\_\_ (fastr.core.samples.SampleIndex attribute), 144  
 \_\_module\_\_ (fastr.core.samples.SampleItem attribute), 145  
 \_\_module\_\_ (fastr.core.samples.SampleItemBase attribute), 145  
 \_\_module\_\_ (fastr.core.samples.SamplePayload attribute), 147  
 \_\_module\_\_ (fastr.core.samples.SampleValue attribute), 147  
 \_\_module\_\_ (fastr.core.serializable.PassThroughSerializer attribute), 147

\_\_module\_\_ (fastr.core.target.LocalBinaryTarget attribute), 150  
 \_\_module\_\_ (fastr.core.target.SystemUsageInfo attribute), 151  
 \_\_module\_\_ (fastr.core.target.Target attribute), 152  
 \_\_module\_\_ (fastr.core.tool.Tool attribute), 153  
 \_\_module\_\_ (fastr.core.toolmanager.ToolManager attribute), 154  
 \_\_module\_\_ (fastr.core.updateable.Updateable attribute), 155  
 \_\_module\_\_ (fastr.core.updateable.UpdateableMeta attribute), 156  
 \_\_module\_\_ (fastr.core.version.Version attribute), 157  
 \_\_module\_\_ (fastr.datatypes.AnalyzeImageFile attribute), 66  
 \_\_module\_\_ (fastr.datatypes.AnyFile attribute), 66  
 \_\_module\_\_ (fastr.datatypes.AnyType attribute), 67  
 \_\_module\_\_ (fastr.datatypes.BaseDataType attribute), 67  
 \_\_module\_\_ (fastr.datatypes.Boolean attribute), 68  
 \_\_module\_\_ (fastr.datatypes.DataType attribute), 69  
 \_\_module\_\_ (fastr.datatypes.Deferred attribute), 69  
 \_\_module\_\_ (fastr.datatypes.Directory attribute), 70  
 \_\_module\_\_ (fastr.datatypes.EnumType attribute), 70  
 \_\_module\_\_ (fastr.datatypes.FilePrefix attribute), 71  
 \_\_module\_\_ (fastr.datatypes.Float attribute), 71  
 \_\_module\_\_ (fastr.datatypes.ITKImageFile attribute), 71  
 \_\_module\_\_ (fastr.datatypes.JsonFile attribute), 71  
 \_\_module\_\_ (fastr.datatypes.MetaImageFile attribute), 72  
 \_\_module\_\_ (fastr.datatypes.NiftiImageFile attribute), 72  
 \_\_module\_\_ (fastr.datatypes.NiftiImageFileCompressed attribute), 72  
 \_\_module\_\_ (fastr.datatypes.NiftiImageFileUncompressed attribute), 72  
 \_\_module\_\_ (fastr.datatypes.Number attribute), 72  
 \_\_module\_\_ (fastr.datatypes.ProvNFile attribute), 73  
 \_\_module\_\_ (fastr.datatypes.String attribute), 73  
 \_\_module\_\_ (fastr.datatypes.TifImageFile attribute), 73  
 \_\_module\_\_ (fastr.datatypes.TxtFile attribute), 73  
 \_\_module\_\_ (fastr.datatypes.TypeGroup attribute), 74  
 \_\_module\_\_ (fastr.datatypes.URLType attribute), 74  
 \_\_module\_\_ (fastr.datatypes.UnsignedInt attribute), 75  
 \_\_module\_\_ (fastr.datatypes.ValueType attribute), 75  
 \_\_module\_\_ (fastr.exceptions.FastrAttributeError attribute), 75  
 \_\_module\_\_ (fastr.exceptions.FastrCannotChangeAttributeError attribute), 75

\_\_module\_\_ (fastr.exceptions.FastrCardinalityError attribute), 75  
\_\_module\_\_ (fastr.exceptions.FastrCollectorError attribute), 76  
\_\_module\_\_ (fastr.exceptions.FastrDataTypeFileNotFoundException attribute), 76  
\_\_module\_\_ (fastr.exceptions.FastrDataTypeMismatchError attribute), 76  
\_\_module\_\_ (fastr.exceptions.FastrDataTypeNotAvailableError attribute), 76  
\_\_module\_\_ (fastr.exceptions.FastrDataTypeNotInstantiableError attribute), 76  
\_\_module\_\_ (fastr.exceptions.FastrDataTypeValueError attribute), 76  
\_\_module\_\_ (fastr.exceptions.FastrError attribute), 76  
\_\_module\_\_ (fastr.exceptions.FastrErrorInSubprocess attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrExecutableNotFoundError attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrExecutionError attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrIOError attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrImportError attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrIndexError attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrIndexNonexistent attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrKeyError attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrLookupError attribute), 77  
\_\_module\_\_ (fastr.exceptions.FastrMountUnknownError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNetworkMismatchError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNetworkUnknownError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNoValidTargetException attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNodeAlreadyPreparedError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNodeNotPreparedError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNodeNotValidError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNotExecutableError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrNotImplementedError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrOSSError attribute), 78  
\_\_module\_\_ (fastr.exceptions.FastrObjectUnknownError attribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrOptionalModuleNotFoundError attribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrOutputValidationAttribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrParentMismatchError attribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrPluginCapabilityNotImplementedAttribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrPluginNotAvailableAttribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrPluginNotLoadedAttribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrResultFileNotFoundException attribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrSerializationError attribute), 79  
\_\_module\_\_ (fastr.exceptions.FastrSerializationIgnoreDefaultError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSerializationInvalidDataError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSerializationMethodError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSinkDataUnavailableError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSizeInvalidError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSizeMismatchError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSizeUnknownError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSourceDataUnavailableError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrStateError attribute), 80  
\_\_module\_\_ (fastr.exceptions.FastrSubprocessNotFinishedError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrToolNotFoundError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrToolTargetNotFoundError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrToolUnknownError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrTypeModelError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrUnknownURLSchemeError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrValueModelError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrVersionInvalidError attribute), 81  
\_\_module\_\_ (fastr.exceptions.FastrVersionMismatchError attribute), 81  
\_\_module\_\_ (fastr.execution.environmentmodules.EnvironmentModulesAttribute), 163  
\_\_module\_\_ (fastr.execution.environmentmodules.ModuleSystemAttribute), 164  
\_\_module\_\_ (fastr.execution.executionpluginmanager.ExecutionPluginAttribute), 166  
\_\_module\_\_ (fastr.execution.executionpluginmanager.ExecutionPluginManagerAttribute), 167  
\_\_module\_\_ (fastr.execution.executionpluginmanager.JobActionAttribute)

attribute), 167  
`__module__` (fastr.execution.job.InlineJob attribute), 168  
`__module__` (fastr.execution.job.Job attribute), 168  
`__module__` (fastr.execution.job.JobState attribute), 172  
`__module__` (fastr.execution.job.SinkJob attribute), 173  
`__module__` (fastr.execution.job.SourceJob attribute), 174  
`__module__` (fastr.execution.networkanalyzer.DefaultNetworkAnalyzer attribute), 174  
`__module__` (fastr.execution.networkanalyzer.NetworkAnalyzer attribute), 174  
`__module__` (fastr.execution.networkchunker.DefaultNetworkChunker attribute), 175  
`__module__` (fastr.execution.networkchunker.NetworkChunker attribute), 175  
`__module__` (fastr.plugins.BlockingExecution attribute), 82  
`__module__` (fastr.plugins.CommaSeperatedValueFile attribute), 83  
`__module__` (fastr.plugins.CrossValidation attribute), 83  
`__module__` (fastr.plugins.DRMAAExecution attribute), 83  
`__module__` (fastr.plugins.FastrInterface attribute), 84  
`__module__` (fastr.plugins.FileSystem attribute), 85  
`__module__` (fastr.plugins.FlowInterface attribute), 86  
`__module__` (fastr.plugins.LinearExecution attribute), 87  
`__module__` (fastr.plugins.NipypeInterface attribute), 87  
`__module__` (fastr.plugins.Null attribute), 88  
`__module__` (fastr.plugins.ProcessPoolExecution attribute), 89  
`__module__` (fastr.plugins.RQExecution attribute), 89  
`__module__` (fastr.plugins.Reference attribute), 89  
`__module__` (fastr.plugins.VirtualFileSystem attribute), 90  
`__module__` (fastr.plugins.VirtualFileSystemRegularExpression attribute), 92  
`__module__` (fastr.plugins.VirtualFileSystemValueList attribute), 93  
`__module__` (fastr.plugins.XNATStorage attribute), 94  
`__module__` (fastr.utils.classproperty.ClassPropertyDescriptor attribute), 178  
`__module__` (fastr.utils.jsonschemaparser.FastrRefResolver attribute), 180  
`__module__` (fastr.utils.pim\_publisher.PimPublisher attribute), 182  
`__module__` (fastr.utils.schematatable.SchemaPrinter attribute), 183  
`__module__` (fastr.web.api.NetworkApi attribute), 187  
`__module__` (fastr.web.api.NetworkListApi attribute), 187  
`__module__` (fastr.web.api.ObjectUrl attribute), 187  
`__module__` (fastr.web.api.Run attribute), 188  
`__module__` (fastr.web.api.RunApi attribute), 188  
`__module__` (fastr.web.api.RunListApi attribute), 188  
`__module__` (fastr.web.api.StatusApi attribute), 188  
`__module__` (fastr.web.api.SubUrl attribute), 189  
`__module__` (fastr.web.api.ToolApi attribute), 189  
`__module__` (fastr.web.api.ToolListApi attribute), 189  
`__ne__()` (fastr.\_\_init\_\_.Network method), 54  
`__ne__()` (fastr.core.network.Network method), 123  
`__ne__()` (fastr.datatypes.BaseDataType method), 67  
`__new__()` (fastr.core.baseplugin.PluginState static method), 99  
`__new__()` (fastr.core.interface.InputSpec static method), 115  
`__new__()` (fastr.core.interface.OutputSpec static method), 116  
`__new__()` (fastr.core.pluginmanager.PluginOptionType static method), 141  
`__new__()` (fastr.core.samples.SampleBaseId static method), 142  
`__new__()` (fastr.core.samples.SampleItem static method), 145  
`__new__()` (fastr.core.samples.SampleItemBase static method), 145  
`__new__()` (fastr.core.samples.SamplePayload static method), 147  
`__new__()` (fastr.core.target.SystemUsageInfo static method), 151  
`__new__()` (fastr.core.updateable.UpdateableMeta static method), 156  
`__new__()` (fastr.core.version.Version static method), 157  
`__new__()` (fastr.datatypes.TypeGroup static method), 74  
`__new__()` (fastr.execution.environmentmodules.ModuleSystem static method), 164  
`__new__()` (fastr.execution.executionpluginmanager.JobAction static method), 167  
`__new__()` (fastr.execution.job.JobState static method), 172  
`__radd__()` (fastr.configmanager.EmptyDefaultExpression method), 66  
`__radd__()` (fastr.core.samples.SampleBaseId method), 143  
`__radd__()` (fastr.core.samples.SampleValue method), 147  
`__reduce_ex__()` (fastr.core.baseplugin.PluginState method), 99  
`__reduce_ex__()` (fastr.datatypes.BaseDataType method), 67  
`__reduce_ex__()` (fastr.datatypes.EnumType method), 70  
`__reduce_ex__()` (fastr.execution.environmentmodules.ModuleSystem method), 164  
`__reduce_ex__()` (fastr.execution.executionpluginmanager.JobAction method), 167  
`__reduce_ex__()` (fastr.execution.job.JobState method), 172  
`__repr__()` (fastr.\_\_init\_\_.Link method), 58  
`__repr__()` (fastr.\_\_init\_\_.Network method), 54

`__repr__()` (fastr.`__init__.Node` method), 60  
`__repr__()` (fastr.configmanager.`Config` method), 65  
`__repr__()` (fastr.core.basemanager.`BaseManager` method), 97  
`__repr__()` (fastr.core.baseplugin.`BasePlugin` method), 98  
`__repr__()` (fastr.core.baseplugin.`PluginMeta` method), 99  
`__repr__()` (fastr.core.baseplugin.`PluginState` method), 99  
`__repr__()` (fastr.core.inputoutput.`BaseInputOutput` method), 104  
`__repr__()` (fastr.core.link.`Link` method), 121  
`__repr__()` (fastr.core.network.`Network` method), 123  
`__repr__()` (fastr.core.node.`Node` method), 133  
`__repr__()` (fastr.core.pluginmanager.`LazyModule` method), 140  
`__repr__()` (fastr.core.pluginmanager.`plugin_option_type` method), 141  
`__repr__()` (fastr.core.samples.`SampleBaseId` method), 143  
`__repr__()` (fastr.core.samples.`SampleCollection` method), 144  
`__repr__()` (fastr.core.samples.`SampleIndex` method), 144  
`__repr__()` (fastr.core.samples.`SampleItemBase` method), 145  
`__repr__()` (fastr.core.samples.`SampleValue` method), 147  
`__repr__()` (fastr.core.target.`SystemUsageInfo` method), 151  
`__repr__()` (fastr.core.tool.`Tool` method), 153  
`__repr__()` (fastr.core.version.`Version` method), 157  
`__repr__()` (fastr.datatypes.`BaseDataType` method), 67  
`__repr__()` (fastr.datatypes.`Deferred` method), 69  
`__repr__()` (fastr.exceptions.`FastrSerializationError` method), 80  
`__repr__()` (fastr.execution.environmentmodules.`EnvironmentModules` method), 163  
`__repr__()` (fastr.execution.environmentmodules.`ModuleSystem` method), 164  
`__repr__()` (fastr.execution.executionpluginmanager.`JobAction` method), 167  
`__repr__()` (fastr.execution.job.`Job` method), 168  
`__repr__()` (fastr.execution.job.`JobState` method), 173  
`__repr__()` (fastr.execution.job.`SinkJob` method), 173  
`__repr__()` (fastr.execution.job.`SourceJob` method), 174  
`__rrshift__()` (fastr.core.inputoutput.`BaseInput` method), 103  
`__schema_type__` (fastr.web.api.`ObjectUrl` attribute), 188  
`__schema_type__` (fastr.web.api.`SubUrl` attribute), 189  
`__setitem__()` (fastr.configmanager.`EmptyDefault` method), 66  
`__setitem__()` (fastr.core.basemanager.`BaseManager` method), 97  
`__setitem__()` (fastr.core.inputoutput.`Input` method), 106  
`__setitem__()` (fastr.core.inputoutput.`Output` method), 109  
`__setitem__()` (fastr.core.inputoutput.`SourceOutput` method), 111  
`__setitem__()` (fastr.core.inputoutput.`SubOutput` method), 114  
`__setitem__()` (fastr.core.node.`InputDict` method), 130  
`__setitem__()` (fastr.core.node.`InputGroup` method), 130  
`__setitem__()` (fastr.core.node.`OutputDict` method), 135  
`__setitem__()` (fastr.core.pluginmanager.`PluginManager` method), 140  
`__setitem__()` (fastr.core.pluginmanager.`PluginsView` method), 141  
`__setitem__()` (fastr.core.samples.`SampleCollection` method), 144  
`__setitem__()` (fastr.core.samples.`SampleValue` method), 147  
`__setstate__()` (fastr.`__init__.ConstantNode` method), 63  
`__setstate__()` (fastr.`__init__.Link` method), 58  
`__setstate__()` (fastr.`__init__.Network` method), 54  
`__setstate__()` (fastr.`__init__.Node` method), 60  
`__setstate__()` (fastr.`__init__.SourceNode` method), 64  
`__setstate__()` (fastr.core.inputoutput.`BaseInputOutput` method), 104  
`__setstate__()` (fastr.core.inputoutput.`Input` method), 106  
`__setstate__()` (fastr.core.inputoutput.`Output` method), 109  
`__setstate__()` (fastr.core.inputoutput.`SubInput` method), 112  
`__setstate__()` (fastr.core.inputoutput.`SubOutput` method), 114  
`__setstate__()` (fastr.core.interface.`Interface` method), 116  
`__setstate__()` (fastr.core.link.`Link` method), 121  
`__setstate__()` (fastr.core.network.`Network` method), 124  
`__setstate__()` (fastr.core.node.`ConstantNode` method), 128  
`__setstate__()` (fastr.core.node.`MacroNode` method), 132  
`__setstate__()` (fastr.core.node.`Node` method), 133  
`__setstate__()` (fastr.core.node.`SinkNode` method), 135  
`__setstate__()` (fastr.core.node.`SourceNode` method), 137  
`__setstate__()` (fastr.core.samples.`SampleValue` method), 147  
`__setstate__()` (fastr.core.tool.`Tool` method), 153  
`__setstate__()` (fastr.core.updateable.`Updateable` method), 155  
`__setstate__()` (fastr.datatypes.`BaseDataType` method), 67  
`__setstate__()` (fastr.datatypes.`Deferred` method), 69  
`__setstate__()` (fastr.execution.job.`Job` method), 168  
`__setstate__()` (fastr.plugins.`FastrInterface` method), 84

`__setstate__(fastr.plugins.FlowInterface method), 86`  
`__setstate__(fastr.plugins.NipypeInterface method), 87`  
`__slotnames__(fastr.web.api.ObjectUrl attribute), 188`  
`__slots__(fastr.core.pluginmanager.plugin_option_type attribute), 141`  
`__slots__(fastr.core.target.SystemUsageInfo attribute), 151`  
`__str__(fastr.__init__.Node method), 61`  
`__str__(fastr.core.baseplugin.BasePlugin method), 98`  
`__str__(fastr.core.baseplugin.PluginState method), 99`  
`__str__(fastr.core.inputoutput.Input method), 106`  
`__str__(fastr.core.inputoutput.Output method), 109`  
`__str__(fastr.core.inputoutput.SubInput method), 112`  
`__str__(fastr.core.inputoutput.SubOutput method), 114`  
`__str__(fastr.core.node.Node method), 133`  
`__str__(fastr.core.samples.SampleBaseId method), 143`  
`__str__(fastr.core.samples.SampleIndex method), 144`  
`__str__(fastr.core.tool.Tool method), 153`  
`__str__(fastr.core.version.Version method), 158`  
`__str__(fastr.datatypes.BaseDataType method), 67`  
`__str__(fastr.datatypes.Boolean method), 68`  
`__str__(fastr.exceptions.FastrError method), 76`  
`__str__(fastr.exceptions.FastrExecutableNotFoundError method), 77`  
`__str__(fastr.exceptions.FastrSerializationError method), 80`  
`__str__(fastr.execution.environmentmodules.ModuleSystem method), 164`  
`__str__(fastr.execution.executionpluginmanager.JobAction method), 167`  
`__str__(fastr.execution.job.JobState method), 173`  
`__str__(fastr.utils.schematatable.SchemaPrinter method), 183`  
`__updatefunc__(fastr.core.node.InputGroup method), 131`  
`__updatefunc__(fastr.core.updateable.Updateable method), 155`  
`__updateinprogress__(fastr.core.updateable.Updateable attribute), 155`  
`__updatetriggers__(fastr.core.node.InputGroup attribute), 131`  
`__updatetriggers__(fastr.core.updateable.Updateable attribute), 155`  
`__updating__(fastr.core.updateable.Updateable attribute), 156`  
`__weakref__(fastr.configmanager.Config attribute), 65`  
`__weakref__(fastr.configmanager.EmptyDefault attribute), 66`  
`__weakref__(fastr.core.baseplugin.BasePlugin attribute), 98`  
`__weakref__(fastr.core.dimension.Dimension attribute), 102`  
`__weakref__(fastr.core.dimension.HasDimensions attribute), 102`  
`__weakref__(fastr.core.interface.InterfaceResult attribute), 116`  
`__weakref__(fastr.core.node.DefaultInputGroupCombiner attribute), 128`  
`__weakref__(fastr.core.pluginmanager.LazyModule attribute), 140`  
`__weakref__(fastr.core.provenance.Provenance attribute), 141`  
`__weakref__(fastr.core.samples.HasSamples attribute), 142`  
`__weakref__(fastr.core.serializable.PassThroughSerializer attribute), 148`  
`__weakref__(fastr.core.serializable.Serializable attribute), 148`  
`__weakref__(fastr.core.target.Target attribute), 152`  
`__weakref__(fastr.core.updateable.Updateable attribute), 156`  
`__weakref__(fastr.exceptions.FastrError attribute), 76`  
`__weakref__(fastr.exceptions.FastrIOError attribute), 77`  
`__weakref__(fastr.exceptions.FastrOSError attribute), 79`  
`__weakref__(fastr.execution.environmentmodules.EnvironmentModules attribute), 163`  
`__weakref__(fastr.execution.networkanalyzer.NetworkAnalyzer attribute), 174`  
`__weakref__(fastr.execution.networkchunker.NetworkChunker attribute), 175`  
`__weakref__(fastr.utils.classproperty.ClassPropertyDescriptor attribute), 178`  
`__weakref__(fastr.utils.pim_publisher.PimPublisher attribute), 182`  
`__weakref__(fastr.utils.schematatable.SchemaPrinter attribute), 183`  
`__weakref__(fastr.web.api.Run attribute), 188`  
`_item_extension()(fastr.core.basemanager.BaseManager method), 96`  
`_load_item()(fastr.core.basemanager.BaseManager method), 96`

## A

`abort()(fastr.__init__.Network method), 54`  
`abort()(fastr.core.network.Network method), 124`  
`abort()(fastr.web.api.Run method), 188`  
`action()(fastr.datatypes.DataType method), 69`  
`action()(fastr.datatypes.Directory method), 70`  
`activity()(fastr.core.provenance.Provenance method), 141`  
`add_link()(fastr.__init__.Network method), 54`  
`add_link()(fastr.core.network.Network method), 124`  
`add_link()(fastr.Network method), 48`  
`add_node()(fastr.__init__.Network method), 55`  
`add_node()(fastr.core.network.Network method), 124`  
`add_node()(fastr.Network method), 48`  
`add_stepid()(fastr.__init__.Network method), 55`  
`add_stepid()(fastr.core.network.Network method), 124`

AdvancedFlowNode (class in fastr.core.node), 127  
AdvancedFlowOutput (class in fastr.core.inputoutput), 102  
agent() (fastr.core.provenance.Provenance method), 141  
aggregate() (fastr.core.target.ProcessUsageCollection method), 151  
analyze\_network() (fastr.execution.networkanalyzer.DefaultNetworkAnalyzer method), 174  
analyze\_network() (fastr.execution.networkanalyzer.NetworkAnalyzer method), 175  
AnalyzeImageFile (class in fastr.datatypes), 66  
any\_of\_draft4() (in module fastr.utils.jsonschemasparser), 180  
AnyFile (class in fastr.datatypes), 66  
AnyType (class in fastr.datatypes), 66  
append() (fastr.configmanager.EmptyDefault method), 66  
append() (fastr.core.inputoutput.Input method), 106  
append() (fastr.core.target.ProcessUsageCollection method), 151  
asdict() (fastr.configmanager.EmptyDefault method), 66  
aslist() (fastr.configmanager.EmptyDefault method), 66  
authors (fastr.core.tool.Tool attribute), 153  
automatic (fastr.core.inputoutput.BaseOutput attribute), 105  
avail() (fastr.execution.environmentmodules.Environment method), 163  
avail\_modules (fastr.execution.environmentmodules.Environment attribute), 163

**B**

BaseDataType (class in fastr.datatypes), 67  
BaseInput (class in fastr.core.inputoutput), 102  
BaseInputOutput (class in fastr.core.inputoutput), 103  
BaseManager (class in fastr.core.basemanager), 96  
basename() (in module fastr.data.url), 159  
BaseOutput (class in fastr.core.inputoutput), 104  
BasePlugin (class in fastr.core.baseplugin), 98  
BasePluginManager (class in module fastr.core.pluginmanager), 139  
blocking (fastr.\_\_init\_\_.Node attribute), 61  
blocking (fastr.core.inputoutput.Output attribute), 109  
blocking (fastr.core.node.FlowNode attribute), 129  
blocking (fastr.core.node.Node attribute), 133  
BlockingExecution (class in fastr.plugins), 82  
Boolean (class in fastr.datatypes), 68  
build (fastr.core.version.Version attribute), 158

**C**

calc\_cardinality() (fastr.execution.job.Job static method), 169  
calcmro() (fastr.core.updateable.UpdateableMeta class method), 156  
call\_subprocess() (fastr.core.target.LocalBinaryTarget method), 150

cancel (fastr.execution.executionpluginmanager.JobAction attribute), 167  
cancel\_job() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
cancelled (fastr.execution.job.JobState attribute), 173  
CANCELS\_DEPENDENCIES (fastr.execution.executionpluginmanager.ExecutionPlugin attribute), 165  
CANCELS\_DEPENDENCIES (fastr.plugins.DRMAAExecution attribute), 83  
cardinality (fastr.core.samples.SampleItemBase attribute), 146  
cardinality() (fastr.\_\_init\_\_.Link method), 58  
cardinality() (fastr.core.inputoutput.BaseInputOutput method), 104  
cardinality() (fastr.core.inputoutput.Input method), 106  
cardinality() (fastr.core.inputoutput.Output method), 109  
cardinality() (fastr.core.inputoutput.SourceOutput method), 111  
cardinality() (fastr.core.inputoutput.SubInput method), 112  
cardinality() (fastr.core.inputoutput.SubOutput method), 114  
cardinality() (fastr.core.link.Link method), 122  
cardinality\_from\_nargs() (in module fastr.utils.cmd.extract\_argparse), 186  
cast() (fastr.core.samples.SampleValue method), 147  
check\_cardinality() (fastr.core.inputoutput.BaseInputOutput method), 104  
check\_finished() (fastr.plugins.RQExecution method), 89

check\_id() (fastr.\_\_init\_\_.Network method), 55  
check\_id() (fastr.core.network.Network method), 124  
check\_input\_id() (fastr.plugins.FastrInterface method), 84  
check\_job\_requirements() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
check\_job\_status() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
check\_output\_id() (fastr.plugins.FastrInterface method), 84  
checksum() (fastr.datatypes.BaseDataType method), 67  
checksum() (fastr.datatypes.Deferred method), 69  
checksum() (fastr.datatypes.FilePrefix method), 71  
checksum() (fastr.datatypes.JsonFile method), 71  
checksum() (fastr.datatypes.MetaImageFile method), 72  
checksum() (fastr.datatypes.URLType method), 74  
checksum() (in module fastr.utils.checksum), 177  
checksum\_directory() (in module fastr.utils.checksum), 177  
chunck\_network() (fastr.execution.networkchunker.DefaultNetworkChunker method), 175  
chunck\_network() (fastr.execution.networkchunker.NetworkChunker method), 175

cite (fastr.core.tool.Tool attribute), 153  
 classproperty() (in module fastr.utils.classproperty), 178  
 ClassPropertyDescriptor (class in fastr.utils.classproperty), 178  
 cleanup() (fastr.core.baseplugin.BasePlugin method), 98  
 cleanup() (fastr.core.ioplugin.IOPlugin method), 117  
 cleanup() (fastr.core.ioplugin.IOPluginManager method), 119  
 cleanup() (fastr.execution.executionpluginmanager.Execution method), 166  
 cleanup() (fastr.plugins.BlockingExecution method), 82  
 cleanup() (fastr.plugins.DRMAAExecution method), 83  
 cleanup() (fastr.plugins.LinearExecution method), 87  
 cleanup() (fastr.plugins.ProcessPoolExecution method), 89  
 cleanup() (fastr.plugins.RQExecution method), 89  
 cleanup() (fastr.plugins.XNATStorage method), 94  
 clear() (fastr.core.inputoutput.Input method), 106  
 clear() (fastr.core.node.InputGroup method), 131  
 clear() (fastr.execution.environmentmodules.Environment method), 163  
 clear\_version() (in module fastr.version), 95  
 collapse (fastr.\_\_init\_\_.Link attribute), 59  
 collapse (fastr.core.link.Link attribute), 122  
 collapse\_indexes (fastr.\_\_init\_\_.Link attribute), 59  
 collapse\_indexes (fastr.core.link.Link attribute), 122  
 collect\_input\_argument\_provenance()  
     (fastr.core.provenance.Provenance method), 141  
 collect\_jobs() (fastr.plugins.DRMAAExecution method), 83  
 collect\_provenance() (fastr.core.provenance.Provenance method), 141  
 collect\_results() (fastr.plugins.FastrInterface method), 84  
 collector\_plugin\_type (fastr.plugins.FastrInterface attribute), 84  
 collectors (fastr.plugins.FastrInterface attribute), 84  
 combine() (fastr.core.samples.SampleItemBase static method), 146  
 command (fastr.core.tool.Tool attribute), 153  
 COMMAND\_DUMP (fastr.execution.job.Job attribute), 168  
 command\_version (fastr.core.tool.Tool attribute), 153  
 commandfile (fastr.execution.job.Job attribute), 169  
 commandurl (fastr.execution.job.Job attribute), 169  
 CommaSeparatedValueFile (class in fastr.plugins), 82  
 compare\_execution\_dir() (in module fastr.utils.compare), 178  
 Config (class in fastr.configmanager), 64  
 configuration\_fields (fastr.core.baseplugin.BasePlugin attribute), 98  
 configuration\_fields (fastr.plugins.DRMAAExecution attribute), 83  
 configuration\_fields (fastr.plugins.ProcessPoolExecution

    attribute), 89  
 configuration\_fields (fastr.plugins.RQExecution attribute), 89  
 connect() (fastr.plugins.XNATStorage method), 94  
 ConstantNode (class in fastr), 50  
 ConstantNode (class in fastr.\_\_init\_\_), 53, 62  
 ConstantNode (class in fastr.core.node), 127  
 container (fastr.core.target.DockerTarget attribute), 150  
 content() (fastr.datatypes.AnalyzeImageFile class method), 66  
 contentPing (fastr.datatypes.MetaImageFile class method), 72  
 content() (fastr.datatypes.URLType class method), 74  
 copy\_file\_dir() (fastr.plugins.VirtualFileSystem static method), 90  
 cpu\_percent (fastr.core.target.SystemUsageInfo attribute), 151  
 create\_constant() (fastr.\_\_init\_\_.Network method), 55  
 create\_constant() (fastr.core.network.Network method), 124  
 create\_enumtype() (fastr.core.datatypemanager.DataTypeManager method), 100  
 ModelIOPlugin\_tool() (fastr.core.ioplugin.IOPluginManager static method), 119  
 create\_job() (fastr.\_\_init\_\_.Node method), 61  
 create\_job() (fastr.\_\_init\_\_.SourceNode method), 64  
 create\_job() (fastr.core.node.Node method), 133  
 create\_job() (fastr.core.node.SinkNode method), 135  
 create\_job() (fastr.core.node.SourceNode method), 137  
 create\_link() (fastr.\_\_init\_\_.Network method), 55  
 create\_link() (fastr.core.network.Network method), 124  
 create\_link() (fastr.Network method), 48  
 create\_link\_from() (fastr.core.inputoutput.BaseInput method), 103  
 create\_macro() (fastr.\_\_init\_\_.Network method), 55  
 create\_macro() (fastr.core.network.Network method), 125  
 create\_macro\_network() (in module fastr.examples.macro\_node), 162  
 create\_network() (in module fastr.examples.add\_ints), 160  
 create\_network() (in module fastr.examples.auto\_prefix), 161  
 create\_network() (in module fastr.examples.collapse), 161  
 create\_network() (in module fastr.examples.collapse\_expand), 161  
 create\_network() (in module fastr.examples.cross\_validation), 161  
 create\_network() (in module fastr.examples.elastix\_nipype), 161  
 create\_network() (in module fastr.examples.expand), 161  
 create\_network() (in module fastr.examples.failing\_network), 162  
 create\_network() (in module fastr.examples.filecopy), 162  
 create\_network() (in module fastr.examples.create\_network())

fastr.examples.macro\_node), 162  
`create_network()` (in module fastr.examples.shift\_links), 162  
`create_network()` (in module fastr.examples.source\_sink), 162  
`create_network_parser()` (in module fastr.utils.cmd.run), 186  
`create_node()` (fastr.`_init__`.Network method), 55  
`create_node()` (fastr.core.network.Network method), 125  
`create_node()` (fastr.Network method), 48  
`create_output_cardinality()` (fastr.core.inputoutput.Output static method), 109  
`create_payload()` (fastr.execution.job.Job method), 169  
`create_payload()` (fastr.execution.job.SinkJob method), 173  
`create_reference()` (fastr.`_init__`.Network method), 56  
`create_reference()` (fastr.core.network.Network method), 125  
`create_reference()` (in module fastr.examples.`_generate_reference__`), 160  
`create_rest_table()` (in module fastr.utils.rest\_generation), 182  
`create_sink()` (fastr.`_init__`.Network method), 56  
`create_sink()` (fastr.core.network.Network method), 125  
`create_sink()` (fastr.Network method), 48  
`create_source()` (fastr.`_init__`.Network method), 56  
`create_source()` (fastr.core.network.Network method), 125  
`create_source()` (fastr.Network method), 49  
`create_super_macro_node()` (in module fastr.examples.macro\_node), 162  
`create vfs_url()` (in module fastr.data.url), 159  
`created` (fastr.execution.job.JobState attribute), 173  
`createobj()` (fastr.`_init__`.Link class method), 59  
`createobj()` (fastr.`_init__`.Node class method), 61  
`createobj()` (fastr.core.link.Link class method), 122  
`createobj()` (fastr.core.node.Node class method), 134  
`createobj()` (fastr.core.serializable.Serializable class method), 148  
CrossValidation (class in fastr.plugins), 83

## D

`data` (fastr.`_init__`.ConstantNode attribute), 63  
`data` (fastr.core.basemanager.BaseManager attribute), 97  
`data` (fastr.core.node.ConstantNode attribute), 128  
`data` (fastr.core.pluginmanager.PluginSubManager attribute), 140  
`data` (fastr.core.samples.SampleItemBase attribute), 146  
`data_uri()` (fastr.core.provenance.Provenance static method), 141  
`DataType` (class in fastr.datatypes), 68  
`datatype` (fastr.`_init__`.SourceNode attribute), 64  
`datatype` (fastr.core.inputoutput.BaseInputOutput attribute), 104  
`datatype` (fastr.core.inputoutput.Input attribute), 106  
`datatype` (fastr.core.inputoutput.Output attribute), 110  
`datatype` (fastr.core.inputoutput.SubOutput attribute), 115  
`datatype` (fastr.core.node.SinkNode attribute), 136  
`datatype` (fastr.core.node.SourceNode attribute), 137  
`datatype_from_type()` (in module fastr.utils.cmd.extract\_argparse), 186  
`DataTypeManager` (class in fastr.core.datatypemanager), 99  
`date_version_matcher` (fastr.core.version.Version attribute), 158  
`debug` (fastr.configmanager.Config attribute), 65  
`DEFAULT_FIELDS` (fastr.configmanager.Config attribute), 64  
`DefaultInputGroupCombiner` (class in fastr.core.node), 128  
`DefaultNetworkAnalyzer` (class in fastr.execution.networkanalyzer), 174  
`DefaultNetworkChunker` (class in fastr.execution.networkchunker), 175  
`Deferred` (class in fastr.datatypes), 69  
`delete()` (fastr.web.api.RunApi method), 188  
`descend()` (fastr.utils.schematable.SchemaPrinter method), 183  
`description` (fastr.core.inputoutput.BaseInputOutput attribute), 104  
`description` (fastr.core.inputoutput.SubInput attribute), 112  
`description` (fastr.core.tool.Tool attribute), 153  
`description` (fastr.datatypes.AnalyzeImageFile attribute), 66  
`description` (fastr.datatypes.AnyFile attribute), 66  
`description` (fastr.datatypes.AnyType attribute), 67  
`description` (fastr.datatypes.BaseDataType attribute), 67  
`description` (fastr.datatypes.Boolean attribute), 68  
`description` (fastr.datatypes.Directory attribute), 70  
`description` (fastr.datatypes.EnumType attribute), 70  
`description` (fastr.datatypes.FilePrefix attribute), 71  
`description` (fastr.datatypes.Float attribute), 71  
`description` (fastr.datatypes.Int attribute), 71  
`description` (fastr.datatypes.ITKImageFile attribute), 71  
`description` (fastr.datatypes.JsonFile attribute), 71  
`description` (fastr.datatypes.MetaImageFile attribute), 72  
`description` (fastr.datatypes.NiftiImageFile attribute), 72  
`description` (fastr.datatypes.NiftiImageFileCompressed attribute), 72  
`description` (fastr.datatypes.NiftiImageFileUncompressed attribute), 72  
`description` (fastr.datatypes.Number attribute), 72  
`description` (fastr.datatypes.ProvNFile attribute), 73  
`description` (fastr.datatypes.String attribute), 73  
`description` (fastr.datatypes.TifImageFile attribute), 73  
`description` (fastr.datatypes.TxtFile attribute), 73

description (fastr.datatypes.UnsignedInt attribute), 75  
 destroy() (fastr.\_\_init\_\_.Link method), 59  
 destroy() (fastr.core.link.Link method), 122  
 dicteq() (in module fastr.utils.dicteq), 178  
 diffdict() (in module fastr.utils.dicteq), 179  
 diffobj() (in module fastr.utils.dicteq), 179  
 diffobj\_str() (in module fastr.utils.dicteq), 179  
 Dimension (class in fastr.core.dimension), 101  
 dimensionality (fastr.core.samples.SampleItemBase attribute), 146  
 dimensions (fastr.core.dimension.HasDimensions attribute), 102  
 dimnames (fastr.\_\_init\_\_.Link attribute), 59  
 dimnames (fastr.\_\_init\_\_.Node attribute), 61  
 dimnames (fastr.\_\_init\_\_.SourceNode attribute), 64  
 dimnames (fastr.core.dimension.HasDimensions attribute), 102  
 dimnames (fastr.core.inputoutput.AdvancedFlowOutput attribute), 102  
 dimnames (fastr.core.inputoutput.Input attribute), 106  
 dimnames (fastr.core.inputoutput.Output attribute), 110  
 dimnames (fastr.core.inputoutput.SubInput attribute), 112  
 dimnames (fastr.core.link.Link attribute), 122  
 dimnames (fastr.core.node.DefaultInputGroupCombiner attribute), 128  
 dimnames (fastr.core.node.FlowNode attribute), 129  
 dimnames (fastr.core.node.InputGroup attribute), 131  
 dimnames (fastr.core.node.Node attribute), 134  
 dimnames (fastr.core.node.SourceNode attribute), 137  
 dimnames (fastr.core.samples.SampleCollection attribute), 144  
 dir\_list() (in module fastr.utils.clear\_pycs), 178  
 Directory (class in fastr.datatypes), 70  
 dirname() (in module fastr.data.url), 159  
 dirurl() (in module fastr.data.url), 159  
 doc() (in module fastr.web.views), 189  
 docker\_api (fastr.core.target.DockerTarget attribute), 150  
 DockerTarget (class in fastr.core.target), 150  
 done (fastr.execution.job.JobState attribute), 173  
 draw\_network() (fastr.\_\_init\_\_.Network method), 56  
 draw\_network() (fastr.core.network.Network method), 125  
 draw\_network() (fastr.Network method), 49  
 DRMAAExecution (class in fastr.plugins), 83  
 dump() (fastr.core.serializable.Serializable method), 148  
 dump() (in module fastr.utils.xmltodict), 185  
 dumpf() (fastr.core.serializable.Serializable method), 148  
 dumpfuncs (fastr.core.serializable.Serializable attribute), 148  
 dumps() (fastr.core.serializable.PassThroughSerializer static method), 148  
 dumps() (fastr.core.serializable.Serializable method), 148  
 dumps() (in module fastr.utils.xmltodict), 185

DYNAMIC\_LIBRARY\_PATH\_DICT  
 (fastr.core.target.LocalBinaryTarget attribute), 150

## E

empty (fastr.core.node.InputGroup attribute), 131  
 EmptyDefault (class in fastr.configmanager), 66  
 endpoint (fastr.web.api.NetworkApi attribute), 187  
 endpoint (fastr.web.api.NetworkListApi attribute), 187  
 endpoint (fastr.web.api.RunApi attribute), 188  
 endpoint (fastr.web.api.RunListApi attribute), 188  
 endpoint (fastr.web.api.StatusApi attribute), 188  
 endpoint (fastr.web.api.ToolApi attribute), 189  
 endpoint (fastr.web.api.ToolListApi attribute), 189  
 ensure\_tmp\_dir() (fastr.execution.job.Job method), 169  
 entity() (fastr.core.provenance.Provenance method), 141  
 EnumType (class in fastr.datatypes), 70  
 EnvironmentModules (class in fastr.execution.environmentmodules), 163  
 envmod (fastr.execution.environmentmodules.ModuleSystem attribute), 164  
 examplesdir (fastr.configmanager.Config attribute), 65  
 exception (fastr.execution.environmentmodules.EnvironmentModules attribute), 163  
 excerpt() (fastr.exceptions.FastrError method), 76  
 exec\_worker() (fastr.plugins.LinearExecution method), 87  
 execute() (fastr.\_\_init\_\_.ConstantNode method), 63  
 execute() (fastr.\_\_init\_\_.Network method), 56  
 execute() (fastr.\_\_init\_\_.Node method), 61  
 execute() (fastr.\_\_init\_\_.SourceNode method), 64  
 execute() (fastr.core.interface.Interface method), 116  
 execute() (fastr.core.network.Network method), 126  
 execute() (fastr.core.node.AdvancedFlowNode method), 127  
 execute() (fastr.core.node.ConstantNode method), 128  
 execute() (fastr.core.node.MacroNode method), 132  
 execute() (fastr.core.node.Node method), 134  
 execute() (fastr.core.node.SinkNode method), 136  
 execute() (fastr.core.node.SourceNode method), 137  
 execute() (fastr.core.tool.Tool method), 153  
 execute() (fastr.execution.job.Job method), 169  
 execute() (fastr.plugins.CrossValidation static method), 83  
 execute() (fastr.plugins.FastrInterface method), 84  
 execute() (fastr.plugins.FlowInterface method), 86  
 execute() (fastr.plugins.NipypeInterface method), 87  
 execute\_job() (in fastr.execution.executionscript module), 167  
 execution\_done (fastr.execution.job.JobState attribute), 173  
 execution\_failed (fastr.execution.job.JobState attribute), 173  
 execution\_plugin (fastr.configmanager.Config attribute), 65  
 ExecutionPlugin (class in fastr.execution.executionpluginmanager),

164  
ExecutionPluginManager (class in fastr.execution.executionpluginmanager), 167  
executionscript (fastr.configmanager.Config attribute), 65  
expand (fastr.\_\_init\_\_.Link attribute), 59  
expand (fastr.core.link.Link attribute), 122  
expand() (fastr.core.samples.SampleIndex method), 145  
expand\_url() (fastr.core.ioplugin.IOPlugin method), 117  
expand\_url() (fastr.core.ioplugin.IOPluginManager method), 119  
expand\_url() (fastr.plugins.CommaSeperatedValueFile method), 83  
expand\_url() (fastr.plugins.VirtualFileSystem method), 90  
expand\_url() (fastr.plugins.VirtualFileSystemRegularExp method), 92  
expand\_url() (fastr.plugins.VirtualFileSystemValueList method), 93  
expand\_url() (fastr.plugins.XNATStorage method), 94  
expanding (fastr.core.interface.Interface attribute), 116  
expanding (fastr.plugins.FastrInterface attribute), 85  
expanding (fastr.plugins.FlowInterface attribute), 87  
expanding (fastr.plugins.NipypeInterface attribute), 88  
extend() (fastr.configmanager.EmptyDefault method), 66  
extend() (in module fastr.utils.jsonschemasparser), 180  
extension (fastr.datatypes.AnalyzeImageFile attribute), 66  
extension (fastr.datatypes.BaseDataType attribute), 67  
extension (fastr.datatypes.Directory attribute), 70  
extension (fastr.datatypes.FilePrefix attribute), 71  
extension (fastr.datatypes.JsonFile attribute), 71  
extension (fastr.datatypes.MetaImageFile attribute), 72  
extension (fastr.datatypes.NiftiImageFileCompressed attribute), 72  
extension (fastr.datatypes.NiftiImageFileUncompressed attribute), 72  
extension (fastr.datatypes.ProvNFile attribute), 73  
extension (fastr.datatypes.TifImageFile attribute), 73  
extension (fastr.datatypes.TxtFile attribute), 73  
extra (fastr.core.version.Version attribute), 158  
extra\_string (fastr.core.version.Version attribute), 158  
extract\_argparser() (in module fastr.utils.cmd.extract\_argparse), 186  
extrainfofile (fastr.execution.job.Job attribute), 169  
extrainfouri (fastr.execution.job.Job attribute), 169

**F**

failed (fastr.core.baseplugin.PluginState attribute), 99  
failed (fastr.execution.job.JobState attribute), 173  
failed\_annotations (fastr.core.samples.SampleItemBase attribute), 146  
fastr.\_\_init\_\_ (module), 53  
fastr.configmanager (module), 64  
fastr.core (module), 96  
fastr.core.basemanager (module), 96  
fastr.core.baseplugin (module), 98  
fastr.core.datatypemanager (module), 99  
fastr.core.dimension (module), 101  
fastr.core.inputoutput (module), 102  
fastr.core.interface (module), 115  
fastr.core.ioplugin (module), 117  
fastr.core.link (module), 120  
fastr.core.network (module), 123  
fastr.core.networkmanager (module), 126  
fastr.core.node (module), 127  
fastr.core.objectmanager (module), 138  
fastr.core.pluginmanager (module), 139  
fastr.core.provenance (module), 141  
fastr.core.samples (module), 142  
fastr.core.serializable (module), 147  
fastr.core.target (module), 150  
fastr.core.test (module), 158  
fastr.core.tool (module), 152  
fastr.core.toolmanager (module), 154  
fastr.core.updateable (module), 155  
fastr.core.version (module), 157  
fastr.core.vfs (module), 158  
fastr.data (module), 159  
fastr.data.url (module), 159  
fastr.datatypes (module), 66  
fastr.examples (module), 160  
fastr.examples.\_\_generate\_reference\_\_ (module), 160  
fastr.examples.add\_ints (module), 160  
fastr.examples.auto\_prefix (module), 161  
fastr.examples.collapse (module), 161  
fastr.examples.collapse\_expand (module), 161  
fastr.examples.cross\_validation (module), 161  
fastr.examples.elastix\_nipype (module), 161  
fastr.examples.expand (module), 161  
fastr.examples.failing\_network (module), 162  
fastr.examples.filecopy (module), 162  
fastr.examples.macro\_node (module), 162  
fastr.examples.shift\_links (module), 162  
fastr.examples.source\_sink (module), 162  
fastr.exceptions (module), 75  
fastr.execution (module), 163  
fastr.execution.environmentmodules (module), 163  
fastr.execution.executionpluginmanager (module), 164  
fastr.execution.executionscript (module), 167  
fastr.execution.job (module), 168  
fastr.execution.networkanalyzer (module), 174  
fastr.execution.networkchunker (module), 175  
fastr.plugins (module), 82  
fastr.resources (module), 175  
fastr.utils (module), 177  
fastr.utils.checksum (module), 177  
fastr.utils.classproperty (module), 178  
fastr.utils.clear\_pycs (module), 178  
fastr.utils.cmd (module), 185  
fastr.utils.cmd.cat (module), 185  
fastr.utils.cmd.execute (module), 186

fastr.utils.cmd.extract\_argparse (module), 186  
 fastr.utils.cmd.prov (module), 186  
 fastr.utils.cmd.run (module), 186  
 fastr.utils.cmd.testtool (module), 186  
 fastr.utils.cmd.trace (module), 186  
 fastr.utils.cmd.verify (module), 187  
 fastr.utils.cmd.webapp (module), 187  
 fastr.utils.compare (module), 178  
 fastr.utils.dicteq (module), 178  
 fastr.utils.gettools (module), 179  
 fastr.utils.iohelpers (module), 179  
 fastr.utils.jsonschema (module), 179  
 fastr.utils.jsonschema.parser (module), 179  
 fastr.utils.multiprocesswrapper (module), 182  
 fastr.utils.pim\_publisher (module), 182  
 fastr.utils.procutils (module), 182  
 fastr.utils.rest\_generation (module), 182  
 fastr.utils.schematable (module), 183  
 fastr.utils.sysinfo (module), 183  
 fastr.utils.verify (module), 184  
 fastr.utils.xmltodict (module), 185  
 fastr.version (module), 95  
 fastr.web (module), 187  
 fastr.web.api (module), 187  
 fastr.web.run (module), 189  
 fastr.web.views (module), 189  
 fastr\_cat() (in module fastr.utils.cmd.cat), 185  
 fastr\_isinstance() (in module fastr.datatypes), 75  
 FastrAttributeError, 75  
 FastrCannotChangeAttributeError, 75  
 FastrCardinalityError, 75  
 FastrCollectorError, 76  
 FastrDataTypeFileNotReadable, 76  
 FastrDataTypeMismatchError, 76  
 FastrDataTypeNotAvailableError, 76  
 FastrDataTypeNotInstantiableError, 76  
 FastrDataTypeValueError, 76  
 FastrError, 76  
 FastrErrorInSubprocess, 77  
 FastrExecutableNotFoundError, 77  
 FastrExecutionError, 77  
 FastrImportError, 77  
 FastrIndexError, 77  
 FastrIndexNonexistent, 77  
 FastrInterface (class in fastr.plugins), 84  
 FastrIOError, 77  
 FastrKeyError, 77  
 FastrLookupError, 77  
 FastrMountUnknownError, 77  
 FastrNetworkMismatchError, 78  
 FastrNetworkUnknownError, 78  
 FastrNodeAlreadyPreparedError, 78  
 FastrNodeNotPreparedError, 78  
 FastrNodeNotValidError, 78  
 FastrNotExecutableError, 78  
 FastrNotImplementedError, 78  
 FastrNoValidTargetError, 78  
 FastrObjectUnknownError, 79  
 FastrOptionalModuleNotAvailableError, 79  
 FastrOSError, 78  
 FastrOutputValidationException, 79  
 FastrParentMismatchError, 79  
 FastrPluginCapabilityNotImplemented, 79  
 FastrPluginNotAvailable, 79  
 FastrPluginNotLoaded, 79  
 FastrRefResolver (class in fastr.utils.jsonschema.parser), 179  
 FastrResultFileNotFoundException, 79  
 FastrSerializationException, 79  
 FastrSerializationIgnoreDefaultError, 80  
 FastrSerializationInvalidDataError, 80  
 FastrSerializationMethodError, 80  
 FastrSinkDataUnavailableError, 80  
 FastrSizeInvalidError, 80  
 FastrSizeMismatchError, 80  
 FastrSizeUnknownError, 80  
 FastrSourceDataUnavailableError, 80  
 FastrStateError, 80  
 FastrSubprocessNotFinished, 81  
 FastrToolNotAvailableError, 81  
 FastrToolTargetNotFound, 81  
 FastrToolUnknownError, 81  
 FastrTypeError, 81  
 FastrUnknownURLSchemeError, 81  
 FastrValueError, 81  
 FastrVersionInvalidError, 81  
 FastrVersionMismatchError, 81  
 fetch\_url() (fastr.core.ioplugin.IOPlugin method), 117  
 fetch\_url() (fastr.plugins.FileSystem method), 85  
 fetch\_url() (fastr.plugins.VirtualFileSystem method), 91  
 fetch\_url() (fastr.plugins.XNATStorage method), 95  
 fetch\_value() (fastr.core.ioplugin.IOPlugin method), 117  
 fetch\_value() (fastr.plugins.FileSystem method), 85  
 fetch\_value() (fastr.plugins.VirtualFileSystem method), 91  
 filename (fastr.core.pluginmanager.plugin\_option\_type attribute), 141  
 filename (fastr.datatypes.AnalyzeImageFile attribute), 66  
 filename (fastr.datatypes.BaseDataType attribute), 68  
 filename (fastr.datatypes.Boolean attribute), 68  
 filename (fastr.datatypes.Directory attribute), 70  
 filename (fastr.datatypes.FilePrefix attribute), 71  
 filename (fastr.datatypes.Float attribute), 71  
 filename (fastr.datatypes.Int attribute), 71  
 filename (fastr.datatypes.ITKImageFile attribute), 71  
 filename (fastr.datatypes.JsonFile attribute), 71  
 filename (fastr.datatypes.MetaImageFile attribute), 72  
 filename (fastr.datatypes.NiftiImageFile attribute), 72  
 filename (fastr.datatypes.NiftiImageFileCompressed attribute), 72  
 filename (fastr.datatypes.NiftiImageFileUncompressed attribute), 72  
 filename (fastr.datatypes.Number attribute), 73  
 filename (fastr.datatypes.ProvNFile attribute), 73

filename (fastr.datatypes.String attribute), 73  
filename (fastr.datatypes.TifImageFile attribute), 73  
filename (fastr.datatypes.TxtFile attribute), 73  
filename (fastr.datatypes.UnsignedInt attribute), 75  
filename (fastr.plugins.BlockingExecution attribute), 82  
filename (fastr.plugins.CommaSepreatedValueFile attribute), 83  
filename (fastr.plugins.CrossValidation attribute), 83  
filename (fastr.plugins.DRMAAExecution attribute), 84  
filename (fastr.plugins.FastrInterface attribute), 85  
filename (fastr.plugins.FileSystem attribute), 86  
filename (fastr.plugins.FlowInterface attribute), 87  
filename (fastr.plugins.LinearExecution attribute), 87  
filename (fastr.plugins.NipypeInterface attribute), 88  
filename (fastr.plugins.Null attribute), 88  
filename (fastr.plugins.ProcessPoolExecution attribute), 89  
filename (fastr.plugins.Reference attribute), 89  
filename (fastr.plugins.RQExecution attribute), 89  
filename (fastr.plugins.VirtualFileSystemRegularExpression attribute), 92  
filename (fastr.plugins.VirtualFileSystemValueList attribute), 93  
filename (fastr.plugins.XNATStorage attribute), 95  
FilePrefix (class in fastr.datatypes), 70  
FileSystem (class in fastr.plugins), 85  
fill\_output\_argument() (fastr.execution.job.Job method), 169  
filter\_plugin() (fastr.core.pluginmanager.PluginsView method), 141  
find\_argparser() (in module fastr.utils.cmd.extract\_argparse), 186  
find\_commands() (in module fastr.utils.cmd), 185  
find\_member() (fastr.core.updateable.UpdateableMeta class method), 156  
find\_source\_index() (fastr.\_\_init\_\_.Node method), 61  
find\_source\_index() (fastr.core.node.InputGroup class method), 131  
find\_source\_index() (fastr.core.node.Node method), 134  
finished (fastr.execution.job.JobState attribute), 173  
Float (class in fastr.datatypes), 71  
flow\_plugin\_type (fastr.plugins.FlowInterface attribute), 87  
flow\_plugins (fastr.plugins.FlowInterface attribute), 87  
FlowInterface (class in fastr.plugins), 86  
FlowNode (class in fastr.core.node), 129  
format() (fastr.web.api.ObjectUrl method), 188  
format() (fastr.web.api.SubUrl method), 189  
from\_schema() (fastr.utils.jsonschema.parser.FastrRefResolver class method), 180  
full\_split() (in module fastr.data.url), 159  
fullid (fastr.\_\_init\_\_.Link attribute), 59  
fullid (fastr.\_\_init\_\_.Network attribute), 57  
fullid (fastr.\_\_init\_\_.Node attribute), 61  
fullid (fastr.core.baseplugin.BasePlugin attribute), 98  
fullid (fastr.core.datatypemanager.DataTypeManager attribute), 100  
fullid (fastr.core.inputoutput.BaseInputOutput attribute), 104  
fullid (fastr.core.inputoutput.Input attribute), 106  
fullid (fastr.core.inputoutput.Output attribute), 110  
fullid (fastr.core.inputoutput.SubInput attribute), 112  
fullid (fastr.core.inputoutput.SubOutput attribute), 115  
fullid (fastr.core.link.Link attribute), 122  
fullid (fastr.core.network.Network attribute), 126  
fullid (fastr.core.node.DefaultInputGroupCombiner attribute), 128  
fullid (fastr.core.node.InputGroup attribute), 131  
fullid (fastr.core.node.Node attribute), 134  
fullid (fastr.core.samples.SampleCollection attribute), 144  
fullid (fastr.core.tool.Tool attribute), 153  
fullid (fastr.datatypes.BaseDataType attribute), 68  
fullid (fastr.execution.job.Job attribute), 169  
function\_wrapper() (in module fastr.utils.multiprocesswrapper), 182

## G

generate\_all\_references() (in module fastr.examples.\_\_generate\_reference\_\_), 160  
get() (fastr.web.api.NetworkApi method), 187  
get() (fastr.web.api.NetworkListApi method), 187  
get() (fastr.web.api.RunApi method), 188  
get() (fastr.web.api.RunListApi method), 188  
get() (fastr.web.api.StatusApi method), 188  
get() (fastr.web.api.ToolApi method), 189  
get() (fastr.web.api.ToolListApi method), 189  
get\_arguments() (fastr.plugins.FastrInterface method), 85  
get\_base\_version() (in module fastr.version), 95  
get\_command\_module() (in module fastr.utils.cmd), 185  
get\_cpu\_usage() (in module fastr.utils.sysinfo), 183  
get\_deferred() (fastr.execution.job.Job method), 169  
get\_drmaa\_info() (in module fastr.utils.sysinfo), 183  
get\_field() (fastr.configmanager.Config method), 65  
get\_hg\_info() (in module fastr.version), 95  
get\_hostinfo() (in module fastr.utils.sysinfo), 183  
get\_job() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
get\_memory\_usage() (in module fastr.utils.sysinfo), 183  
get\_mounts() (in module fastr.utils.sysinfo), 184  
get\_object\_version() (fastr.core.networkmanager.NetworkManager method), 127  
get\_object\_version() (fastr.core.objectmanager.ObjectManager method), 138  
get\_object\_version() (fastr.core.toolmanager.ToolManager method), 154  
get\_os() (in module fastr.utils.sysinfo), 184  
get\_output\_datatype() (fastr.execution.job.Job method), 169

get\_output\_datatype() (fastr.execution.job.SourceJob method), 174  
 get\_parent\_provenance() (fastr.core.provenance.Provenance static method), 142  
 get\_parser() (in module fastr.utils.cmd.cat), 185  
 get\_parser() (in module fastr.utils.cmd.execute), 186  
 get\_parser() (in module fastr.utils.cmd.extract\_argparse), 186  
 get\_parser() (in module fastr.utils.cmd.prov), 186  
 get\_parser() (in module fastr.utils.cmd.run), 186  
 get\_parser() (in module fastr.utils.cmd.testtool), 186  
 get\_parser() (in module fastr.utils.cmd.trace), 186  
 get\_parser() (in module fastr.utils.cmd.verify), 187  
 get\_parser() (in module fastr.utils.cmd.webapp), 187  
 get\_path\_from\_url() (in module fastr.data.url), 159  
 get\_processes() (in module fastr.utils.sysinfo), 184  
 get\_prov\_document() (in module fastr.utils.cmd.prov), 186  
 get\_python() (in module fastr.utils.sysinfo), 184  
 get\_result() (fastr.execution.job.InlineJob method), 168  
 get\_result() (fastr.execution.job.Job method), 170  
 get\_result() (fastr.execution.job.SinkJob method), 173  
 get\_saved\_version() (in module fastr.version), 95  
 get\_serializer() (fastr.core.serializable.Serializable class method), 149  
 get\_sourced\_nodes() (fastr.\_\_init\_\_.Node method), 61  
 get\_sourced\_nodes() (fastr.core.inputoutput.Input method), 106  
 get\_sourced\_nodes() (fastr.core.inputoutput.SubInput method), 112  
 get\_sourced\_nodes() (fastr.core.node.Node method), 134  
 get\_sourced\_outputs() (fastr.core.inputoutput.Input method), 107  
 get\_sourced\_outputs() (fastr.core.inputoutput.SubInput method), 113  
 get\_specials() (fastr.plugins.FastrInterface method), 85  
 get\_status() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
 get\_subinput() (fastr.core.inputoutput.Input method), 107  
 get\_sysinfo() (in module fastr.utils.sysinfo), 184  
 get\_type() (fastr.core.datatypemanager.DataTypeManager method), 100  
 get\_type() (fastr.plugins.NipypeInterface method), 88  
 get\_url\_scheme() (in module fastr.data.url), 160  
 get\_users() (in module fastr.utils.sysinfo), 184  
 get\_value() (fastr.execution.job.Job class method), 170  
 getblueprinter() (in module fastr.utils.jsonschemasparser), 180  
 guess\_type() (fastr.core.datatypemanager.DataTypeManager method), 100

hash (fastr.core.tool.Tool attribute), 153  
 hash\_inputs() (fastr.execution.job.Job method), 170  
 hash\_inputs() (fastr.execution.job.SinkJob method), 173  
 hash\_inputs() (fastr.execution.job.SourceJob method), 174  
 hash\_results() (fastr.execution.job.Job method), 170  
 hashsum() (in module fastr.utils.checksum), 177  
 HasSamples (class in fastr.core.samples), 142  
 help (fastr.core.tool.Tool attribute), 153  
 hold (fastr.execution.executionpluginmanager.JobAction attribute), 167  
 hold (fastr.execution.job.JobState attribute), 173  
 hold\_job() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166

---

**H**

has\_type() (fastr.core.datatypemanager.DataTypeManager method), 100  
 HasDimensions (class in fastr.core.dimension), 102

id (fastr.\_\_init\_\_.Network attribute), 57  
 id (fastr.\_\_init\_\_.Node attribute), 61  
 id (fastr.core.baseplugin.BasePlugin attribute), 98  
 id (fastr.core.inputoutput.BaseInputOutput attribute), 104  
 id (fastr.core.network.Network attribute), 126  
 id (fastr.core.node.Node attribute), 134  
 id (fastr.core.pluginmanager.plugin\_option\_type attribute), 141  
 id (fastr.core.samples.SampleItemBase attribute), 146  
 id (fastr.datatypes.BaseDataType attribute), 68  
 id (fastr.execution.job.Job attribute), 170  
 id (fastr.execution.job.SinkJob attribute), 173  
 id\_(fastr.core.node.Node attribute), 134  
 id\_(fastr.Node attribute), 49  
 ids() (fastr.core.samples.HasSamples method), 142  
 in\_progress (fastr.execution.job.JobState attribute), 173  
 index (fastr.core.samples.SampleItemBase attribute), 146  
 index() (fastr.core.inputoutput.Input method), 107  
 index() (in module fastr.web.views), 189  
 indexPlugin (fastr.core.samples.HasSamples method), 142  
 indexrep (fastr.core.inputoutput.SubOutput attribute), 115  
 INFO\_DUMP (fastr.execution.job.Job attribute), 168  
 init\_provenance() (fastr.core.provenance.Provenance method), 142  
 InlineJob (class in fastr.execution.job), 168  
 Input (class in fastr.core.inputoutput), 105  
 input (fastr.core.node.SinkNode attribute), 136  
 input\_group (fastr.core.inputoutput.Input attribute), 107  
 input\_group (fastr.core.inputoutput.SubInput attribute), 113  
 input\_groups (fastr.\_\_init\_\_.Node attribute), 61  
 input\_groups (fastr.core.node.DefaultInputGroupCombiner attribute), 128  
 input\_groups (fastr.core.node.Node attribute), 134  
 InputDict (class in fastr.core.node), 129  
 InputGroup (class in fastr.core.node), 130  
 inputs (fastr.core.interface.Interface attribute), 116  
 inputs (fastr.core.node.Node attribute), 134

inputs (fastr.core.tool.Tool attribute), 153  
inputs (fastr.Node attribute), 50  
inputs (fastr.plugins.FastrInterface attribute), 85  
inputs (fastr.plugins.FlowInterface attribute), 87  
inputs (fastr.plugins.NipypeInterface attribute), 88  
InputSpec (class in fastr.core.interface), 115  
InputSpecBase (in module fastr.core.interface), 115  
insert() (fastr.core.inputoutput.Input method), 107  
instantiate (fastr.core.baseplugin.BasePlugin attribute), 98  
Int (class in fastr.datatypes), 71  
Interface (class in fastr.core.interface), 115  
interface\_class (fastr.core.tool.Tool attribute), 153  
InterfacePluginManager (class in fastr.core.interface), 116  
InterfaceResult (class in fastr.core.interface), 116  
IOPlugin (class in fastr.core.ioplugin), 117  
IOPluginManager (class in fastr.core.ioplugin), 119  
is\_mapping (fastr.core.samples.SampleValue attribute), 147  
is\_sequence (fastr.core.samples.SampleValue attribute), 147  
is\_valid() (fastr.\_\_init\_\_.Network method), 57  
is\_valid() (fastr.core.network.Network method), 126  
isdatatype() (fastr.core.datatypemanager.DataTypeManager static method), 101  
isinstance() (fastr.datatypes.BaseDataType class method), 68  
isinstance() (fastr.datatypes.TypeGroup class method), 74  
isloaded() (fastr.execution.environmentmodules.EnvironmentModules method), 163  
isslice (fastr.core.samples.SampleIndex attribute), 145  
isurl() (fastr.core.ioplugin.IOPlugin static method), 117  
isurl() (in module fastr.data.url), 160  
items() (fastr.core.samples.HasSamples method), 142  
items\_prevalidate() (in module fastr.utils.jsonschema.parser), 180  
iter\_input\_groups() (fastr.core.node.DefaultInputGroup method), 128  
iter\_input\_groups() (fastr.core.node.MergingInputGroup method), 132  
iterconvergingindices() (fastr.core.inputoutput.Output method), 110  
iterelements() (fastr.core.samples.SampleValue method), 147  
iterinputvalues (fastr.core.node.InputGroup attribute), 131  
iteritems() (fastr.core.inputoutput.SubInput method), 113  
iteritems() (fastr.core.samples.HasSamples method), 142  
itersubinputs() (fastr.core.inputoutput.BaseInput method), 103  
itersubinputs() (fastr.core.inputoutput.Input method), 107  
itersubinputs() (fastr.core.inputoutput.SubInput method), 113  
ITKImageFile (class in fastr.datatypes), 71

## J

Job (class in fastr.execution.job), 168  
job (fastr.datatypes.Deferred attribute), 69  
job\_finished() (fastr.\_\_init\_\_.Network method), 57  
job\_finished() (fastr.core.network.Network method), 126  
job\_finished() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
JobAction (class in fastr.execution.executionpluginmanager), 167  
jobs (fastr.core.samples.SampleItemBase attribute), 146  
JobState (class in fastr.execution.job), 171  
join() (in module fastr.data.url), 160  
json (in module fastr.plugins), 95  
JsonFile (class in fastr.datatypes), 71

## L

LazyModule (class in fastr.core.pluginmanager), 139  
LinearExecution (class in fastr.plugins), 87  
linearized (fastr.core.inputoutput.SourceOutput attribute), 111  
Link (class in fastr), 49  
Link (class in fastr.\_\_init\_\_), 53, 57  
Link (class in fastr.core.link), 120  
listeners (fastr.\_\_init\_\_.Node attribute), 61  
listeners (fastr.core.inputoutput.Output attribute), 110  
listeners (fastr.core.inputoutput.SubOutput attribute), 115  
listeners (fastr.core.node.Node attribute), 134  
lmod (fastr.execution.environmentmodules.ModuleSystem attribute), 164  
load() (fastr.core.serializable.Serializable class method), 149  
load() (fastr.execution.environmentmodules.EnvironmentModules method), 163  
load() (in module fastr.utils.xmltodict), 185  
load\_gpickle() (in module fastr.utils.iohelpers), 179  
load\_json() (in module fastr.utils.iohelpers), 179  
load\_plugin() (fastr.core.pluginmanager.BasePluginManager method), 139  
loaded (fastr.core.baseplugin.PluginState attribute), 99  
loaded\_modules (fastr.execution.environmentmodules.EnvironmentModules attribute), 163  
loadf() (fastr.core.serializable.Serializable class method), 149  
loads() (fastr.core.serializable.PassThroughSerializer static method), 148  
loads() (fastr.core.serializable.Serializable class method), 149  
loads() (in module fastr.utils.xmltodict), 185  
LocalBinaryTarget (class in fastr.core.target), 150  
logdir (fastr.configmanager.Config attribute), 65  
logfile (fastr.execution.job.Job attribute), 170  
logtype (fastr.configmanager.Config attribute), 65  
logurl (fastr.execution.job.Job attribute), 170

lookup() (fastr.datatypes.Deferred class method), 69

## M

MacroNode (class in fastr.core.node), 132  
 main() (in module fastr.core.ioplugin), 120  
 main() (in module fastr.examples.add\_ints), 160  
 main() (in module fastr.examples.auto\_prefix), 161  
 main() (in module fastr.examples.collapse), 161  
 main() (in module fastr.examples.collapse\_expand), 161  
 main() (in module fastr.examples.cross\_validation), 161  
 main() (in module fastr.examples.elastix\_nipype), 161  
 main() (in module fastr.examples.expand), 161  
 main() (in module fastr.examples.failing\_network), 162  
 main() (in module fastr.examples.filecopy), 162  
 main() (in module fastr.examples.macro\_node), 162  
 main() (in module fastr.examples.shift\_links), 162  
 main() (in module fastr.examples.source\_sink), 162  
 main() (in module fastr.execution.executionscript), 167  
 main() (in module fastr.utils.clear\_pycs), 178  
 main() (in module fastr.utils.cmd), 185  
 main() (in module fastr.utils.cmd.cat), 185  
 main() (in module fastr.utils.cmd.execute), 186  
 main() (in module fastr.utils.cmd.extract\_argparse), 186  
 main() (in module fastr.utils.cmd.prov), 186  
 main() (in module fastr.utils.cmd.run), 186  
 main() (in module fastr.utils.cmd.testtool), 186  
 main() (in module fastr.utils.cmd.trace), 186  
 main() (in module fastr.utils.cmd.verify), 187  
 main() (in module fastr.utils.cmd.webapp), 187  
 main() (in module fastr.utils.getools), 179  
 main() (in module fastr.web.run), 189  
 major (fastr.core.version.Version attribute), 158  
 mapping\_part() (fastr.core.samples.SampleValue method), 147  
 match\_filename() (fastr.core.basemanager.BaseManager method), 97  
 match\_types() (fastr.core.datatypemanager.DataTypeManager method), 101  
 match\_types\_any() (fastr.core.datatypemanager.DataTypeManager method), 101  
 md5\_checksum() (in module fastr.utils.checksum), 177  
 mediatypes() (fastr.web.api.NetworkApi method), 187  
 mediatypes() (fastr.web.api.NetworkListApi method), 187  
 mediatypes() (fastr.web.api.RunApi method), 188  
 mediatypes() (fastr.web.api.RunListApi method), 188  
 mediatypes() (fastr.web.api.StatusApi method), 188  
 mediatypes() (fastr.web.api.ToolApi method), 189  
 mediatypes() (fastr.web.api.ToolListApi method), 189  
 members (fastr.datatypes.TypeGroup attribute), 74  
 merge() (fastr.core.node.DefaultInputGroupCombiner method), 128  
 merge() (fastr.core.node.MergingInputGroupCombiner method), 132  
 merge\_dimensions (fastr.\_\_init\_\_.Node attribute), 61  
 merge\_dimensions (fastr.core.node.Node attribute), 134  
 merge\_failed\_annotations() (fastr.core.node.DefaultInputGroupCombiner method), 128  
 merge\_payloads() (fastr.core.node.DefaultInputGroupCombiner method), 129  
 merge\_sample\_data() (fastr.core.node.DefaultInputGroupCombiner method), 129  
 merge\_sample\_id() (fastr.core.node.DefaultInputGroupCombiner method), 129  
 merge\_sample\_index() (fastr.core.node.DefaultInputGroupCombiner method), 129  
 merge\_sample\_jobs() (fastr.core.node.DefaultInputGroupCombiner method), 129  
 MergingInputGroupCombiner (class in fastr.core.node), 132  
 messages (fastr.core.updateable.Updateable attribute), 156  
 MetaImageFile (class in fastr.datatypes), 71  
 methods (fastr.web.api.NetworkApi attribute), 187  
 methods (fastr.web.api.NetworkListApi attribute), 187  
 methods (fastr.web.api.RunApi attribute), 188  
 methods (fastr.web.api.RunListApi attribute), 188  
 methods (fastr.web.api.StatusApi attribute), 189  
 methods (fastr.web.api.ToolApi attribute), 189  
 methods (fastr.web.api.ToolListApi attribute), 189  
 minor (fastr.core.version.Version attribute), 158  
 module (fastr.core.baseplugin.BasePlugin attribute), 98  
 module (fastr.datatypes.AnalyzeImageFile attribute), 66  
 module (fastr.datatypes.Boolean attribute), 68  
 module (fastr.datatypes.Directory attribute), 70  
 module (fastr.datatypes.FilePrefix attribute), 71  
 module (fastr.datatypes.Float attribute), 71  
 module (fastr.datatypes.Int attribute), 71  
 module (fastr.datatypes.ITKImageFile attribute), 71  
 module (fastr.datatypes.JsonFile attribute), 71  
 module (fastr.datatypes.MetaImageFile attribute), 72  
 module (fastr.datatypes.NiftiImageFile attribute), 72  
 module (fastr.datatypes.NiftiImageFileCompressed attribute), 72  
 module (fastr.datatypes.NiftiImageFileUncompressed attribute), 72  
 module (fastr.datatypes.Number attribute), 73  
 module (fastr.datatypes.ProvNFile attribute), 73  
 module (fastr.datatypes.String attribute), 73  
 module (fastr.datatypes.TifImageFile attribute), 73  
 module (fastr.datatypes.TxtFile attribute), 73  
 module (fastr.datatypes.UnsignedInt attribute), 75  
 module (fastr.plugins.BlockingExecution attribute), 82  
 module (fastr.plugins.CommaSeperatedValueFile attribute), 83  
 module (fastr.plugins.CrossValidation attribute), 83  
 module (fastr.plugins.DRMAAExecution attribute), 84  
 module (fastr.plugins.FastrInterface attribute), 85  
 module (fastr.plugins.FileSystem attribute), 86  
 module (fastr.plugins.FlowInterface attribute), 87  
 module (fastr.plugins.LinearExecution attribute), 87

module (fastr.plugins.NipypeInterface attribute), 88  
module (fastr.plugins.Null attribute), 88  
module (fastr.plugins.ProcessPoolExecution attribute), 89  
module (fastr.plugins.Reference attribute), 89  
module (fastr.plugins.RQExecution attribute), 89  
module (fastr.plugins.VirtualFileSystemRegularExpression attribute), 92  
module (fastr.plugins.VirtualFileSystemValueList attribute), 93  
module (fastr.plugins.XNATStorage attribute), 95  
ModuleSystem (class in fastr.execution.environmentmodules), 164  
monitor\_docker() (fastr.core.target.DockerTarget method), 150  
monitor\_process() (fastr.core.target.LocalBinaryTarget method), 151  
mounts (fastr.configmanager.Config attribute), 65

## N

name (fastr.\_\_init\_\_.Node attribute), 61  
name (fastr.ConstantNode attribute), 50  
name (fastr.core.node.Node attribute), 134  
name (fastr.core.pluginmanager.plugin\_option\_type attribute), 141  
name (fastr.core.tool.Tool attribute), 153  
name (fastr.datatypes.BaseDataType attribute), 68  
name (fastr.Node attribute), 50  
name (fastr.SinkNode attribute), 50  
name (fastr.SourceNode attribute), 50  
namedtuple\_to\_dict() (in module fastr.utils.sysinfo), 184  
namespace (fastr.core.network.Network attribute), 126  
namespace (fastr.core.pluginmanager.plugin\_option\_type attribute), 141  
namespace (fastr.core.tool.Tool attribute), 153  
ndims (fastr.core.inputoutput.Output attribute), 110  
ndims (fastr.core.inputoutput.SourceOutput attribute), 111  
ndims (fastr.core.samples.SampleCollection attribute), 144  
Network (class in fastr), 48  
Network (class in fastr.\_\_init\_\_), 53, 54  
Network (class in fastr.core.network), 123  
network() (in module fastr.web.views), 189  
NETWORK\_DUMP\_FILE\_NAME (fastr.\_\_init\_\_.Network attribute), 54  
NETWORK\_DUMP\_FILE\_NAME (fastr.core.network.Network attribute), 123  
network\_lock\_thread() (in module fastr.web.api), 189  
network\_runner() (in module fastr.web.api), 189  
NetworkAnalyzer (class in fastr.execution.networkanalyzer), 174  
NetworkApi (class in fastr.web.api), 187  
NetworkChunker (class in fastr.execution.networkchunker), 175  
networklist (in module fastr.core.networkmanager), 127

NetworkListApi (class in fastr.web.api), 187  
NetworkManager (class in fastr.core.networkmanager), 126  
networks() (in module fastr.web.views), 189  
networks\_path (fastr.configmanager.Config attribute), 65

NiftiImageFile (class in fastr.datatypes), 72  
NiftiImageFileCompressed (class in fastr.datatypes), 72  
NiftiImageFileUncompressed (class in fastr.datatypes), 72

NipypeInterface (class in fastr.plugins), 87  
Node (class in fastr), 49  
Node (class in fastr.\_\_init\_\_), 53, 60  
Node (class in fastr.core.node), 132  
node (fastr.core.inputoutput.BaseInputOutput attribute), 104  
node (fastr.core.inputoutput.SubInput attribute), 113  
node (fastr.core.inputoutput.SubOutput attribute), 115  
node\_class (fastr.core.tool.Tool attribute), 154  
nodegroup (fastr.\_\_init\_\_.Node attribute), 61  
nodegroup (fastr.core.node.Node attribute), 134  
nonexistent (fastr.execution.job.JobState attribute), 173  
normurl() (in module fastr.data.url), 160  
not\_draft4() (in module fastr.utils.jsonschemasparser), 181

ns\_id (fastr.core.tool.Tool attribute), 154  
Null (class in fastr.plugins), 88  
num\_subinput (fastr.core.inputoutput.BaseInput attribute), 103  
num\_subinput (fastr.core.inputoutput.Input attribute), 107  
num\_subinput (fastr.core.inputoutput.SubInput attribute), 113

Number (class in fastr.datatypes), 72  
numel (fastr.core.inputoutput.BaseInputOutput attribute), 104

## O

object\_class (fastr.core.networkmanager.NetworkManager attribute), 127  
object\_class (fastr.core.objectmanager.ObjectManager attribute), 138  
object\_class (fastr.core.toolmanager.ToolManager attribute), 154  
ObjectManager (class in fastr.core.objectmanager), 138  
ObjectUrl (class in fastr.web.api), 187  
objectversions() (fastr.core.objectmanager.ObjectManager method), 139

one\_of\_draft4() (in module fastr.utils.jsonschemasparser), 181  
open\_url() (in module fastr.utils.cmd.webapp), 187  
options (fastr.datatypes.EnumType attribute), 70

Output (class in fastr.core.inputoutput), 108  
output (fastr.\_\_init\_\_.SourceNode attribute), 64  
output (fastr.core.node.SourceNode attribute), 137  
OutputDict (class in fastr.core.node), 135  
outputs (fastr.core.interface.Interface attribute), 116  
outputs (fastr.core.node.Node attribute), 134

outputs (fastr.core.tool.Tool attribute), 154  
 outputs (fastr.Node attribute), 50  
 outputs (fastr.plugins.FastrInterface attribute), 85  
 outputs (fastr.plugins.FlowInterface attribute), 87  
 outputs (fastr.plugins.NipypeInterface attribute), 88  
 outputsize (fastr.\_\_init\_\_.Node attribute), 61  
 outputsize (fastr.\_\_init\_\_.SourceNode attribute), 64  
 outputsize (fastr.core.node.DefaultInputGroupCombiner attribute), 129  
 outputsize (fastr.core.node.FlowNode attribute), 129  
 outputsize (fastr.core.node.Node attribute), 134  
 outputsize (fastr.core.node.SourceNode attribute), 137  
 OutputSpec (class in fastr.core.interface), 116  
 OutputSpecBase (in module fastr.core.interface), 117

**P**

parent (fastr.\_\_init\_\_.Link attribute), 59  
 parent (fastr.\_\_init\_\_.Node attribute), 61  
 parent (fastr.core.link.Link attribute), 122  
 parent (fastr.core.node.InputGroup attribute), 131  
 parent (fastr.core.node.Node attribute), 134  
 parent (fastr.core.samples.SampleCollection attribute), 144  
 parent (fastr.datatypes.BaseDataType attribute), 68  
 parse() (fastr.utils.schematatable.SchemaPrinter method), 183  
 parsed\_value (fastr.datatypes.BaseDataType attribute), 68  
 parsed\_value (fastr.datatypes.Deferred attribute), 69  
 parsed\_value (fastr.datatypes.URLType attribute), 74  
 PassThroughSerializer (class in fastr.core.serializable), 147  
 path (fastr.core.tool.Tool attribute), 154  
 path (in module fastr.plugins), 95  
 path\_to\_url() (fastr.core.ioplugin.IOPlugin method), 117  
 path\_to\_url() (fastr.plugins.FileSystem method), 86  
 path\_to\_url() (fastr.plugins.VirtualFileSystem method), 91  
 pattern\_properties\_prevail() (in module fastr.utils.jsonschemasparser), 181  
 pim\_host (fastr.configmanager.Config attribute), 65  
 pim\_register\_run() (fastr.utils.pim\_publisher.PimPublisher method), 182  
 pim\_serialize\_network() (fastr.utils.pim\_publisher.PimPublisher static method), 182  
 PIM\_STATUS\_MAPPING (fastr.utils.pim\_publisher.PimPublisher attribute), 182  
 pim\_update\_status() (fastr.utils.pim\_publisher.PimPublisher method), 182  
 PimPublisher (class in fastr.utils.pim\_publisher), 182  
 Plugin (class in fastr.core.baseplugin), 99  
 plugin\_class (fastr.core.datatypemanager.DataTypeManager attribute), 101  
 plugin\_class (fastr.core.pluginmanager.BasePluginManager attribute), 139  
 plugin\_class (fastr.core.pluginmanager.PluginManager attribute), 140  
 plugin\_class (fastr.core.pluginmanager.PluginSubManager attribute), 140  
 plugin\_option\_type (class in fastr.core.pluginmanager), 141  
 PluginManager (class in fastr.core.pluginmanager), 140  
 PluginMeta (class in fastr.core.baseplugin), 99  
 plugins\_path (fastr.configmanager.Config attribute), 65  
 PluginState (class in fastr.core.baseplugin), 99  
 PluginSubManager (class in fastr.core.pluginmanager), 140  
 PluginsView (class in fastr.core.pluginmanager), 140  
 poll\_datatype() (fastr.core.datatypemanager.DataTypeManager method), 101  
 pop() (fastr.core.node.InputGroup method), 131  
 popitem() (fastr.core.node.InputGroup method), 131  
 populate() (fastr.core.basemanager.BaseManager method), 97  
 populate() (fastr.core.datatypemanager.DataTypeManager method), 101  
 populate() (fastr.core.ioplugin.IOPluginManager method), 119  
 populate() (fastr.core.pluginmanager.BasePluginManager method), 139  
 populate() (fastr.core.toolmanager.ToolManager method), 155  
 post() (fastr.web.api.RunListApi method), 188  
 preferred\_types (fastr.configmanager.Config attribute), 65  
 preferred\_types (fastr.core.inputoutput.Output attribute), 110  
 preferred\_types (fastr.core.inputoutput.SubOutput attribute), 115  
 preload (fastr.core.baseplugin.PluginState attribute), 99  
 prepare() (fastr.\_\_init\_\_.Node method), 62  
 prepare() (fastr.core.inputoutput.Input method), 107  
 prepare() (fastr.core.inputoutput.Output method), 110  
 prepare() (fastr.core.node.Node method), 134  
 prepend() (fastr.configmanager.EmptyDefault method), 66  
 primary (fastr.core.node.InputGroup attribute), 131  
 print\_help() (in module fastr.utils.cmd), 185  
 print\_job\_result() (in module fastr.utils.cmd.trace), 186  
 print\_result() (fastr.core.ioplugin.IOPlugin static method), 118  
 print\_sample\_node() (in module fastr.utils.cmd.trace), 186  
 print\_sample\_sink() (in module fastr.utils.cmd.trace), 186  
 print\_samples() (in module fastr.utils.cmd.trace), 187  
 print\_sinks() (in module fastr.utils.cmd.trace), 187  
 printlines() (fastr.utils.schematatable.SchemaPrinter method), 183  
 process\_pool\_worker\_number (fastr.configmanager.Config attribute), 65  
 processing\_callback (fastr.execution.job.JobState at-

tribute), 173  
ProcessPoolExecution (class in fastr.plugins), 88  
ProcessUsageCollection (class in fastr.core.target), 151  
properties\_postvalidate() (in module fastr.utils.jsonschema.parser), 181  
properties\_prevalidate() (in module fastr.utils.jsonschema.parser), 181  
protected\_modules (fastr.configmanager.Config attribute), 65  
prov() (in module fastr.web.views), 189  
PROV\_DUMP (fastr.execution.job.Job attribute), 168  
Provenance (class in fastr.core.provenance), 141  
provenance (fastr.datatypes.Deferred attribute), 69  
provfile (fastr.execution.job.Job attribute), 170  
ProvNFile (class in fastr.datatypes), 73  
provurl (fastr.execution.job.Job attribute), 170  
pull\_source\_data() (fastr.core.ioplugin.IOPlugin method), 118  
pull\_source\_data() (fastr.core.ioplugin.IOPluginManager method), 119  
push\_sink\_data() (fastr.core.ioplugin.IOPlugin method), 118  
push\_sink\_data() (fastr.core.ioplugin.IOPluginManager method), 119  
push\_sink\_data() (fastr.plugins.Reference method), 89  
put\_url() (fastr.core.ioplugin.IOPlugin method), 118  
put\_url() (fastr.core.ioplugin.IOPluginManager method), 120  
put\_url() (fastr.plugins.FileSystem method), 86  
put\_url() (fastr.plugins.Null method), 88  
put\_url() (fastr.plugins.VirtualFileSystem method), 91  
put\_url() (fastr.plugins.XNATStorage method), 95  
put\_value() (fastr.core.ioplugin.IOPlugin method), 118  
put\_value() (fastr.plugins.FileSystem method), 86  
put\_value() (fastr.plugins.Null method), 88  
put\_value() (fastr.plugins.VirtualFileSystem method), 91

Python Enhancement Proposals  
PEP 8, 21  
PEP 8#class-names, 21  
PEP 8#global-variable-names, 21  
PEP 8#method-names-and-instance-variables, 21  
PEP 8#package-and-module-names, 21  
PEP 8#prescriptive-naming-conventions, 21

**Q**

queue (fastr.execution.executionpluginmanager.JobAction attribute), 167  
queue\_job() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
queued (fastr.execution.job.JobState attribute), 173

**R**

raw\_value (fastr.datatypes.BaseDataType attribute), 68  
read\_bytes (fastr.core.target.SystemUsageInfo attribute), 151  
read\_config() (fastr.configmanager.Config method), 65  
read\_config\_files (fastr.configmanager.Config attribute), 65  
read\_sink\_data() (in module fastr.utils.cmd.trace), 187  
readfastrschema() (fastr.utils.jsonschema.parser.FastrRefResolver static method), 180  
readfile() (fastr.utils.jsonschema.parser.FastrRefResolver static method), 180  
ready (fastr.core.updateable.Updateable attribute), 156  
Reference (class in fastr.plugins), 89  
references (fastr.core.tool.Tool attribute), 154  
regex (fastr.core.tool.Tool attribute), 154  
register\_configuration() (fastr.core.baseplugin.BasePlugin class method), 98  
register\_fields() (fastr.configmanager.Config method), 65  
register\_job() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
register\_url\_scheme() (fastr.core.ioplugin.IOPluginManager static method), 120  
register\_url\_scheme() (in module fastr.data.url), 160  
release\_job() (fastr.execution.executionpluginmanager.ExecutionPlugin method), 166  
reload() (fastr.core.basemanager.BaseManager method), 97  
reload() (fastr.execution.environmentmodules.EnvironmentModules method), 163  
remove() (fastr.\_\_init\_\_.Network method), 57  
remove() (fastr.core.inputoutput.Input method), 107  
remove() (fastr.core.inputoutput.SubInput method), 113  
remove() (fastr.core.network.Network method), 126  
request\_parser (fastr.web.api.RunListApi attribute), 188  
required (fastr.core.inputoutput.BaseInputOutput attribute), 104  
required\_cores (fastr.\_\_init\_\_.Node attribute), 62  
required\_cores (fastr.core.node.Node attribute), 134  
required\_cores (fastr.execution.job.Job attribute), 170  
required\_memory (fastr.\_\_init\_\_.Node attribute), 62  
required\_memory (fastr.core.node.Node attribute), 134  
required\_memory (fastr.execution.job.Job attribute), 170  
required\_time (fastr.\_\_init\_\_.Node attribute), 62  
required\_time (fastr.core.node.Node attribute), 134  
required\_time (fastr.execution.job.Job attribute), 170  
requirements (fastr.core.tool.Tool attribute), 154  
resourcesdir (fastr.configmanager.Config attribute), 65  
RESULT\_DUMP (fastr.execution.job.Job attribute), 168  
ExecutionPlugin  
resulting\_datatype (fastr.core.inputoutput.Output attribute), 110  
resulting\_datatype (fastr.core.inputoutput.SubOutput attribute), 115  
rmem (fastr.core.target.SystemUsageInfo attribute), 151  
RQExecution (class in fastr.plugins), 89  
Run (class in fastr.web.api), 188  
run\_command() (fastr.core.target.DockerTarget

method), 150  
`run_command()` (fastr.core.target.LocalBinaryTarget method), 151  
`run_command()` (fastr.core.target.Target method), 152  
`run_id` (fastr.\_\_init\_\_.Network attribute), 57  
`run_id` (fastr.core.network.Network attribute), 126  
`run_job()` (fastr.plugins.RQExecution class method), 89  
`run_network()` (fastr.web.api.Run method), 188  
`RunApi` (class in fastr.web.api), 188  
`runapp()` (in module fastr.web.run), 189  
`RunListApi` (class in fastr.web.api), 188  
`running` (fastr.execution.job.JobState attribute), 173

**S**

`SampleBaseId` (class in fastr.core.samples), 142  
`SampleCollection` (class in fastr.core.samples), 143  
`SampleId` (class in fastr.core.samples), 144  
`SampleIndex` (class in fastr.core.samples), 144  
`SampleItem` (class in fastr.core.samples), 145  
`SampleItemBase` (class in fastr.core.samples), 145  
`SamplePayload` (class in fastr.core.samples), 146  
`samples` (fastr.core.inputoutput.Output attribute), 110  
`samples` (fastr.core.inputoutput.SubOutput attribute), 115  
`SampleValue` (class in fastr.core.samples), 147  
`save_gpickle()` (in module fastr.utils.iohelpers), 179  
`save_json()` (in module fastr.utils.iohelpers), 179  
`save_version()` (in module fastr.version), 95  
`schemadir` (fastr.configmanager.Config attribute), 65  
`SchemaPrinter` (class in fastr.utils.schematable), 183  
`scheme` (fastr.core.ioplugin.IOPlugin attribute), 118  
`scheme` (fastr.plugins.CommaSeparatedValueFile attribute), 83  
`scheme` (fastr.plugins.FileSystem attribute), 86  
`scheme` (fastr.plugins.Null attribute), 88  
`scheme` (fastr.plugins.Reference attribute), 90  
`scheme` (fastr.plugins.VirtualFileSystem attribute), 91  
`scheme` (fastr.plugins.VirtualFileSystemRegularExpression attribute), 92  
`scheme` (fastr.plugins.VirtualFileSystemValueList attribute), 93  
`scheme` (fastr.plugins.XNATStorage attribute), 95  
`send_job()` (fastr.plugins.DRMAAExecution method), 84  
`sequence_part()` (fastr.core.samples.SampleValue method), 147  
`Serializable` (class in fastr.core.serializable), 148  
`serialize()` (fastr.core.provenance.Provenance method), 142  
`SERIALIZERS` (fastr.core.serializable.Serializable attribute), 148  
`server` (fastr.plugins.XNATStorage attribute), 95  
`set_code()` (fastr.core.baseplugin.BasePlugin class method), 98  
`set_data()` (fastr.\_\_init\_\_.ConstantNode method), 63  
`set_data()` (fastr.\_\_init\_\_.SourceNode method), 64  
`set_data()` (fastr.core.node.ConstantNode method), 128  
`set_data()` (fastr.core.node.SinkNode method), 136  
`set_data()` (fastr.core.node.SourceNode method), 138  
`set_field()` (fastr.configmanager.Config method), 65  
`set_result()` (fastr.\_\_init\_\_.Node method), 62  
`set_result()` (fastr.core.node.AdvancedFlowNode method), 127  
`set_result()` (fastr.core.node.FlowNode method), 129  
`set_result()` (fastr.core.node.Node method), 135  
`set_result()` (fastr.core.node.SinkNode method), 136  
`set_status()` (fastr.core.baseplugin.BasePlugin class method), 98  
`set_subinput()` (fastr.core.inputoutput.Input method), 107  
`setdefault()` (fastr.core.node.InputGroup method), 131  
`setup()` (fastr.core.ioplugin.IOPlugin method), 118  
`setup()` (fastr.plugins.VirtualFileSystem method), 91  
`sha1_checksum()` (in module fastr.utils.checksum), 177  
`show_jobs()` (fastr.execution.executionpluginmanager.ExecutionPlugin method), 167  
`shutdown()` (in module fastr.web.views), 189  
`shutdown_server()` (in module fastr.web.views), 190  
`sink_data()` (in module fastr.examples.add\_ints), 160  
`sink_data()` (in module fastr.examples.collapse), 161  
`sink_data()` (in module fastr.examples.collapse\_expand), 161  
`sink_data()` (in module fastr.examples.cross\_validation), 161  
`sink_data()` (in module fastr.examples.elastix\_nipype), 161  
`sink_data()` (in module fastr.examples.expand), 161  
`sink_data()` (in module fastr.examples.failing\_network), 162  
`sink_data()` (in module fastr.examples.filecopy), 162  
`sink_data()` (in module fastr.examples.macro\_node), 162  
`sink_data()` (in module fastr.examples.shift\_links), 162  
`sink_data()` (in module fastr.examples.source\_sink), 162  
`SINK_DUMP_FILE_NAME` (fastr.\_\_init\_\_.Network attribute), 54  
`SINK_DUMP_FILE_NAME` (fastr.core.network.Network attribute), 123  
`SinkJob` (class in fastr.execution.job), 173  
`SinkNode` (class in fastr), 50  
`SinkNode` (class in fastr.\_\_init\_\_), 53  
`SinkNode` (class in fastr.core.node), 135  
`size` (fastr.\_\_init\_\_.Link attribute), 59  
`size` (fastr.core.dimension.HasDimensions attribute), 102  
`size` (fastr.core.inputoutput.BaseInputOutput attribute), 104  
`size` (fastr.core.inputoutput.Input attribute), 108  
`size` (fastr.core.inputoutput.Output attribute), 110  
`size` (fastr.core.inputoutput.SourceOutput attribute), 111  
`size` (fastr.core.inputoutput.SubInput attribute), 113  
`size` (fastr.core.link.Link attribute), 122  
`size` (fastr.core.node.InputGroup attribute), 131

size (fastr.core.samples.HasSamples attribute), 142  
size (fastr.core.samples.SampleCollection attribute), 144  
solve\_broadcast() (fastr.core.node.InputGroup class method), 132  
source (fastr.\_\_init\_\_.Link attribute), 59  
source (fastr.core.inputoutput.Input attribute), 108  
source (fastr.core.inputoutput.SubInput attribute), 113  
source (fastr.core.link.Link attribute), 122  
source (fastr.Link attribute), 49  
source\_code (fastr.core.baseplugin.BasePlugin attribute), 99  
source\_data() (in module fastr.examples.add\_ints), 160  
source\_data() (in module fastr.examples.collapse), 161  
source\_data() (in module fastr.examples.collapse\_expand), 161  
source\_data() (in module fastr.examples.cross\_validation), 161  
source\_data() (in module fastr.examples.elastix\_nipype), 161  
source\_data() (in module fastr.examples.expand), 161  
source\_data() (in module fastr.examples.failing\_network), 162  
source\_data() (in module fastr.examples.filecopy), 162  
source\_data() (in module fastr.examples.macro\_node), 162  
source\_data() (in module fastr.examples.shift\_links), 162  
source\_data() (in module fastr.examples.source\_sink), 162  
SOURCE\_DUMP\_FILE\_NAME (fastr.\_\_init\_\_.Network attribute), 54  
SOURCE\_DUMP\_FILE\_NAME (fastr.core.network.Network attribute), 123  
source\_job\_limit (fastr.configmanager.Config attribute), 65  
source\_output (fastr.core.inputoutput.SubInput attribute), 113  
sourcegroup (fastr.\_\_init\_\_.SourceNode attribute), 64  
sourcegroup (fastr.core.node.SourceNode attribute), 138  
SourceJob (class in fastr.execution.job), 174  
SourceNode (class in fastr), 50  
SourceNode (class in fastr.\_\_init\_\_), 53, 63  
SourceNode (class in fastr.core.node), 136  
SourceOutput (class in fastr.core.inputoutput), 110  
split() (in module fastr.data.url), 160  
status (fastr.\_\_init\_\_.Link attribute), 60  
status (fastr.\_\_init\_\_.Node attribute), 62  
status (fastr.core.baseplugin.BasePlugin attribute), 99  
status (fastr.core.link.Link attribute), 123  
status (fastr.core.node.Node attribute), 135  
status (fastr.core.version.Version attribute), 158  
status (fastr.execution.job.Job attribute), 170  
status() (fastr.web.api.Run method), 188  
status\_message (fastr.core.baseplugin.BasePlugin attribute), 99  
StatusApi (class in fastr.web.api), 188  
STDERR\_DUMP (fastr.execution.job.Job attribute), 168  
stderrfile (fastr.execution.job.Job attribute), 170  
stderrurl (fastr.execution.job.Job attribute), 170  
stdout (in module fastr.plugins), 95  
STDOUT\_DUMP (fastr.execution.job.Job attribute), 168  
stdoutfile (fastr.execution.job.Job attribute), 170  
stdouturl (fastr.execution.job.Job attribute), 170  
String (class in fastr.datatypes), 73  
SubInput (class in fastr.core.inputoutput), 111  
submit\_jobs() (fastr.plugins.DRMAAExecution method), 84  
SubOutput (class in fastr.core.inputoutput), 113  
substitute() (fastr.execution.job.SinkJob method), 173  
SubUrl (class in fastr.web.api), 189  
suffix (fastr.core.version.Version attribute), 158  
SUPPORTS\_CANCEL (fastr.execution.executionpluginmanager.ExecutionPlugin attribute), 165  
SUPPORTS\_CANCEL (fastr.plugins.DRMAAExecution attribute), 83  
SUPPORTS\_DEPENDENCY (fastr.execution.executionpluginmanager.ExecutionPlugin attribute), 165  
SUPPORTS\_DEPENDENCY (fastr.plugins.DRMAAExecution attribute), 83  
SUPPORTS\_HOLD\_RELEASE (fastr.execution.executionpluginmanager.ExecutionPlugin attribute), 165  
SUPPORTS\_HOLD\_RELEASE (fastr.plugins.DRMAAExecution attribute), 83  
swap() (fastr.execution.environmentmodules.EnvironmentModules method), 163  
switch\_sample\_sink() (in module fastr.utils.cmd.trace), 187  
sync() (fastr.execution.environmentmodules.EnvironmentModules method), 164  
systemdir (fastr.configmanager.Config attribute), 65  
SystemUsageInfo (class in fastr.core.target), 151

## T

tags (fastr.core.tool.Tool attribute), 154  
Target (class in fastr.core.target), 151  
target (fastr.\_\_init\_\_.Link attribute), 60  
target (fastr.core.link.Link attribute), 123  
target (fastr.core.tool.Tool attribute), 154  
target (fastr.datatypes.Deferred attribute), 69  
target (fastr.Link attribute), 49  
test() (fastr.\_\_init\_\_.Network method), 57  
test() (fastr.core.baseplugin.BasePlugin class method), 99  
test() (fastr.core.interface.Interface class method), 116  
test() (fastr.core.network.Network method), 126

test() (fastr.core.tool.Tool method), 154  
 test() (fastr.datatypes.BaseDataType class method), 68  
 test() (fastr.plugins.DRMAAExecution class method), 84  
 test() (fastr.plugins.NipypeInterface class method), 88  
 test() (fastr.plugins.ProcessPoolExecution class method), 89  
 test\_spec (fastr.core.tool.Tool attribute), 154  
 tests (fastr.core.tool.Tool attribute), 154  
 TifImageFile (class in fastr.datatypes), 73  
 timestamp (fastr.core.target.SystemUsageInfo attribute), 151  
 tmpurl (fastr.execution.job.Job attribute), 170  
 tmpurl (fastr.execution.job.SinkJob attribute), 174  
 todict() (fastr.core.objectmanager.ObjectManager method), 139  
 Tool (class in fastr.core.tool), 152  
 tool (fastr.\_\_init\_\_.Node attribute), 62  
 tool (fastr.core.node.Node attribute), 135  
 tool (fastr.execution.job.Job attribute), 170  
 tool() (in module fastr.web.views), 190  
 ToolApi (class in fastr.web.api), 189  
 toollist (fastr attribute), 47  
 toollist (in module fastr.\_\_init\_\_), 53  
 toollist (in module fastr.core.toolmanager), 155  
 ToolListApi (class in fastr.web.api), 189  
 ToolManager (class in fastr.core.toolmanager), 154  
 tools\_path (fastr.configmanager.Config attribute), 65  
 toolversions() (fastr.core.toolmanager.ToolManager method), 155  
 tostring\_modvalue() (fastr.execution.environmentmodules static method), 164  
 totuple\_modvalue() (fastr.execution.environmentmodules static method), 164  
 translate\_argument() (fastr.execution.job.Job method), 171  
 translate\_results() (fastr.execution.job.Job method), 171  
 TxtFile (class in fastr.datatypes), 73  
 TypeGroup (class in fastr.datatypes), 73  
 typelist (fastr attribute), 48  
 typelist (in module fastr.\_\_init\_\_), 53  
 types\_path (fastr.configmanager.Config attribute), 65

**U**

uninitialized (fastr.core.baseplugin.PluginState attribute), 99  
 unload() (fastr.execution.environmentmodules.Environment method), 164  
 unloaded (fastr.core.baseplugin.PluginState attribute), 99  
 unmerge() (fastr.core.node.DefaultInputGroupCombiner method), 129  
 unmerge() (fastr.core.node.MergingInputGroupCombiner method), 132  
 UnsignedInt (class in fastr.datatypes), 74  
 update() (fastr.configmanager.EmptyDefault method), 66

update() (fastr.core.node.DefaultInputGroupCombiner method), 129  
 update() (fastr.core.node.InputGroup method), 132  
 update() (fastr.core.node.MergingInputGroupCombiner method), 132  
 update() (fastr.core.updateable.Updateable method), 156  
 update\_input\_groups() (fastr.\_\_init\_\_.Node method), 62  
 update\_input\_groups() (fastr.core.node.Node method), 135  
 update\_job\_result() (in module fastr.web.api), 189  
 update\_status() (in module fastr.web.api), 189  
 Updateable (class in fastr.core.updateable), 155  
 UpdateableMeta (class in fastr.core.updateable), 156  
 updatetrigger() (fastr.core.updateable.UpdateableMeta static method), 157  
 url (fastr.core.tool.Tool attribute), 154  
 url\_to\_path() (fastr.core.ioplugin.IOPlugin method), 119  
 url\_to\_path() (fastr.core.ioplugin.IOPluginManager method), 120  
 url\_to\_path() (fastr.plugins.FileSystem method), 86  
 url\_to\_path() (fastr.plugins.VirtualFileSystem method), 91  
 URLType (class in fastr.datatypes), 74  
 usage\_type (fastr.core.target.ProcessUsageCollection attribute), 151  
 userdir (fastr.configmanager.Config attribute), 65

**V** EnvironmentModules

valid (fastr.\_\_init\_\_.SourceNode attribute), 64  
 valid (fastr.core.environmentmodules.EnvironmentModule output.Output attribute), 110  
 valid (fastr.core.node.SourceNode attribute), 138  
 valid (fastr.core.updateable.Updateable attribute), 156  
 valid (fastr.datatypes.BaseDataType attribute), 68  
 valid (fastr.datatypes.URLType attribute), 74  
 validate\_results() (fastr.execution.job.Job method), 171  
 validate\_results() (fastr.execution.job.SinkJob method), 174  
 validate\_results() (fastr.execution.job.SourceJob method), 174  
 value (fastr.datatypes.BaseDataType attribute), 68  
 value (fastr.datatypes.Boolean attribute), 68  
 value (fastr.datatypes.Deferred attribute), 70  
 value (fastr.datatypes.Float attribute), 71  
 value (fastr.datatypes.Int attribute), 71  
 value (fastr.datatypes.UnsignedInt attribute), 75  
 ValueType (class in fastr.datatypes), 75  
 verify\_resource\_loading() (in module fastr.utils.verify), 184  
 verify\_tool() (in module fastr.utils.verify), 184  
 Version (class in fastr.core.version), 157  
 version (fastr.core.tool.Tool attribute), 154  
 version (fastr.datatypes.BaseDataType attribute), 68  
 version (fastr.datatypes.EnumType attribute), 70  
 version\_matcher (fastr.core.version.Version attribute), 158

VirtualFileSystem (class in fastr.plugins), [90](#)  
VirtualFileSystemRegularExpression (class in fastr.plugins), [91](#)  
VirtualFileSystemValueList (class in fastr.plugins), [92](#)  
vmem (fastr.core.target.SystemUsageInfo attribute), [151](#)

## W

warn\_develop (fastr.configmanager.Config attribute), [65](#)  
web\_hostname (fastr.configmanager.Config attribute), [65](#)  
web\_port (fastr.configmanager.Config attribute), [65](#)  
web\_secret\_key (fastr.configmanager.Config attribute), [65](#)  
web\_url() (fastr.configmanager.Config method), [65](#)  
websocket\_client() (in module fastr.web.views), [190](#)  
which() (in module fastr.utils.procutils), [182](#)  
write() (fastr.execution.job.Job method), [171](#)  
write\_bytes (fastr.core.target.SystemUsageInfo attribute), [151](#)

## X

x (fastr.configmanager.Config attribute), [65](#)  
xnat (fastr.plugins.XNATStorage attribute), [95](#)  
XNATStorage (class in fastr.plugins), [93](#)